Tactical Cloudlets Webinar

Table of Contents

Carnegie Mellon University Notices

Carnegie Mellon University

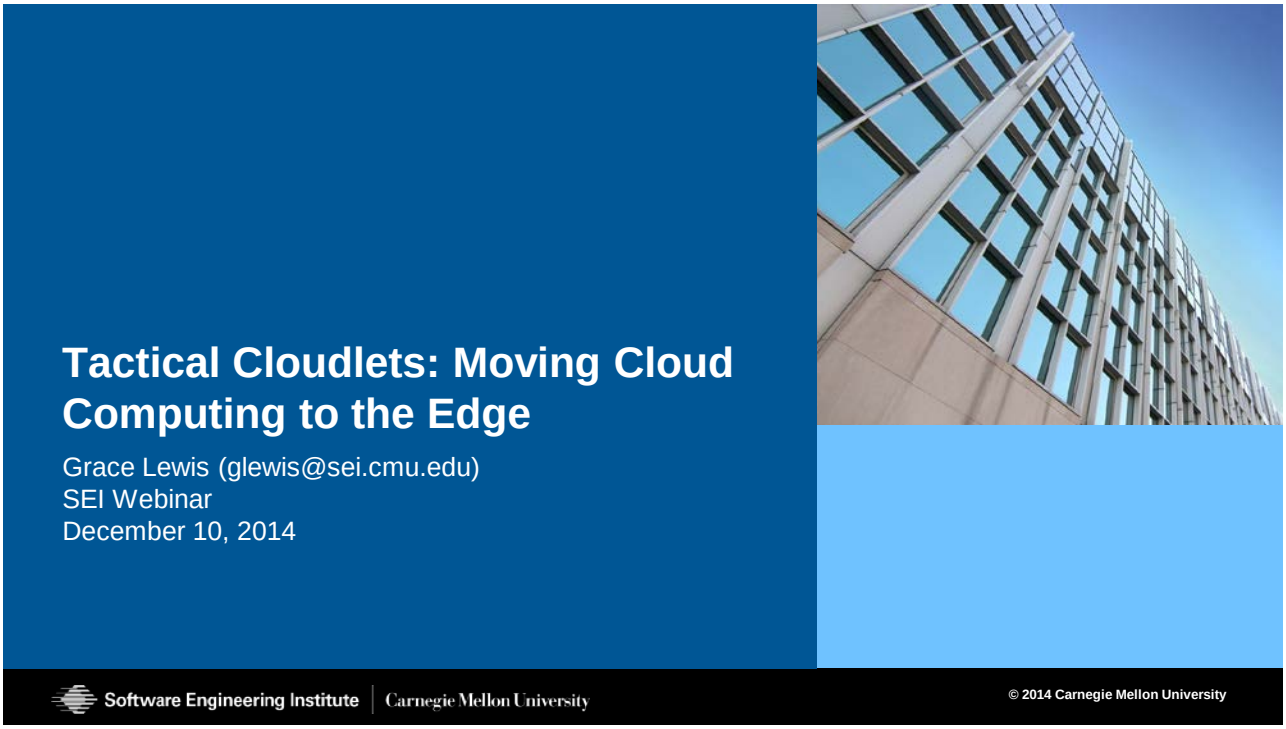
This video and all related information and materials ("materials") are owned by Carnegie Mellon University. These materials are provided on an "as-is" "as available" basis without any warranties and solely for your personal viewing and use.

You agree that Carnegie Mellon is not liable with respect to any materials received by you as a result of viewing the video, or using referenced websites, and/or for any consequences or the use by you of such materials.

By viewing, downloading, and/or using this video and related materials, you agree that you have read and agree to our terms of use (www.sei.cmu.edu/legal/).

DM-0001955

**002 (Music)



**001 Shane McGraw: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to the Software Engineering Institute's Webinar Series.

Our presentation today is Tactical Cloudlets: Moving Cloud Computing to the Edge, by Grace Lewis.

Depending on your location, we wish you a good morning, a good afternoon or a good evening.

My name is Shane McGraw, your moderator for today, and I'd like to thank you for attending.

We want to make today as interactive as possible. So we will take questions throughout the

presentation and again at the end of the presentation. You can submit your questions to our Events Staff at any time through the Questions tab on your Control Panel.

We will also ask a few polling questions throughout the presentation. They will appear as a popup window on your screen.

Another three tabs I'd like to point out are the Files, Twitter and Survey tabs. The File tab has a copy of the PDF- a PDF copy of the presentation slides there now, along with other work from the SEI in Mobile Computing.

The Survey tab will appear at the end of the presentation and we ask that you fill out as your feedback is always greatly appreciated.

And for those of you using Twitter, be sure to follow @seinews and use the hash tag seicloud; once again, follow @seinews and the hash tag is seicloud.

Now I'd like to introduce our presenter for today.

Grace Lewis is a principal researcher at the Software Engineering Institute at Carnegie Mellon University. She's a Deputy Lead for the Advanced Mobile Systems Initiative and the Principal Investigator for the Edge-Enabled Projects- for Edge-Enabled Tactical Systems Research Project.

Her current interests and projects are in Mobile Computing, Cloud Computing and Service-Oriented Architecture.

Her latest publications include multiple papers and articles on these subjects in a book in the SEI Software Engineering Series.

She is also a member of the Technical Faculty for the Masters in Software Engineering Program at Carnegie Mellon University.

And now I'd like to turn it over to Grace Lewis. Grace, welcome, all yours.

Motivation

Motivation

Soldiers, first responders and field personnel operating in tactical environments increasingly make use of handheld devices to help with tasks such as face recognition, language translation, decision-making, and mission planning.

Edge environments are characterized by dynamic context, limited computing resources, high levels of stress, and intermittent network connectivity.

Tactical cloudlets provide cloud capabilities at the edge that can lead to enhanced situational awareness and decision making, even if disconnected from the enterprise.

Software Engineering Institute | Carnegie Mellon University

Tactical Cloudlets
Grace A. Lewis — Dec. 10, 2014

3

**003 Grace Lewis: Thank you, thank you so much.

What I am going to talk about today is some of the research that we've been doing in an area called Tactical Cloudlets.

I would start- I would like to start this-- it's more- it's going to be more like a story because I'm going to be telling you about how we started with this research, where we are now and where we're going into the future.

So the motivation for this research is that soldiers, first responders and field personnel that operate in tactical environments are making increased use of applications such as speech recognition, such as facial recognition, mission planning; a lot of applications that take a real heavy toll on computing power and battery power.

And if you think about the environments in which these people operate, they're what we called edge environments. They're environments with a very dynamic context; because, for example, if you're in an emergency rescue situation, you could be in a period of I want to say peace at a moment but then something could happen, some natural event could happen.

They're also limited computing resources. Why? Because handheld devices, it doesn't matter how much we advance in computing, handheld devices are always going to lag behind their desktop counterparts.

They're also high levels of stress; if you think of the environments in which these people operate.

And finally there's intermittent network connectivity just because they're operating at the edge.

So what we're trying to do in our research is to come up with these tactical cloudlets. You can think of them as data centers in a box; and the idea is they provide cloud capabilities at the edge that can lead to better situational awareness and also better decision making, even if they're disconnected from the enterprise.

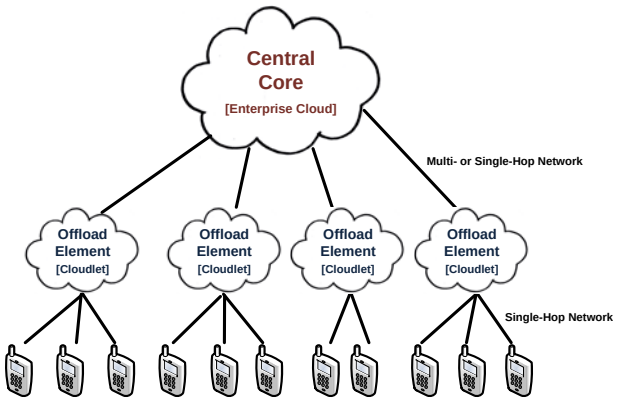
Tactical Cloudlets 1

Tactical Cloudlets 1

Goal: Provide cloud computing capabilities at the edge for computation offload, data staging, and increased survivability of mobile systems

R&D Goals

- Discoverable resources
- Operation in DIL environments (disconnected, intermittent, limited)
- Systems perspective on cyber-foraging that includes survivability, trust, and ease of development and deployment
- Flexible architecture to support research and experimentation



**004 So as I said before, the goal behind this research is to provide cloud computing capabilities at the edge. For what? For things such as computation offload. I'm talking about offload of expensive

computation, for data staging, and eventually leading to increased survivability of mobile systems.

From a research and development perspective what we worked hard getting with tactical cloudlets was discoverable resources. We wanted these cloudlets to be located in the field and for them to be discovered by the smartphones of personnel operating in the field.

We wanted them to be able to operate in what is now called DIL environments; DIL stands for Disconnected, Intermittent and Limited, which means that there is not always connectivity-- if you look at the diagram on the right side of the slide-- there is not always going to be connectivity between those cloudlets or offload elements and the central core enterprise cloud.

We also wanted to take a systems perspective. Based on a systematic literature review that we did in this area, there is a lot of work in cloudlets and in cyber-foraging, which is one of the main topics I'm going to talk about today; and this work is very interesting, it's very promising, but it's really focusing on the offload operations, how complex it is, the algorithms.

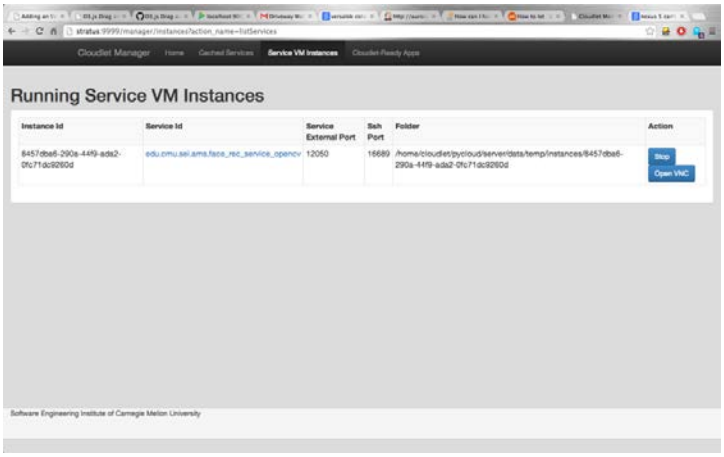
We're making sure that the offload operation happens and we really want to take more of a systems perspective. We want to think about survivability, we want to think about trust, we want to think about how

easy it would be to develop for these environments and also to deploy, because we're thinking about fielded operational systems.

And finally, as part of our research we wanted to create a very flexible architecture that would support our research and our experimentation.

Tactical Cloudlets 2

Tactical Cloudlets 2



Capabilities: Forward-deployed, discoverable, virtual machine (VM) cloudlets that can be hosted on vehicles or other platforms and provide

- infrastructure to offload computation
- forward data-staging for a mission
- data filtering to remove unnecessary data from streams intended for mobile users
- collection points for data heading for enterprise repositories

**005 So on this slide you will see a screen from our technical cloudlet implementation. It's the Cloudlet Manager screen; and I will talk about it later. But basically what-- if I were to define tactical cloudlets I would define them as forward deployed, discoverable, virtual machine-based; and I'm talking about servers.

When I say server, it's the laptop, it could be a larger computer. But they're basically forward deployed. They are deployed in proximity of the people who use them. They can be hosted on vehicles; they can be hosted on other platforms.

And what do tactical clouds provide? They provide an infrastructure in which to offload computation. They provide forward data staging for our missions.

So for example, if I know that I'm eventually going to use certain data on a mission, I can use these cloudlets to store this data so it can be available for me when I need it.

I can also use cloudlets to do data filtering. Let's say I'm receiving a lot of information from the cloud or from the enterprise or from the data center. I could have these cloudlets do some pre-filtering on data such that on the mobile device I don't receive a large amount of information but I receive really what I just need.

And finally, I can also use them in the opposite direction, meaning I can use cloudlets as collection points for data heading to enterprise repositories.

So for example, I am capturing data at the edge, I'm offloading it on to these cloudlets and these cloudlets they're serving basically as intermediaries for data heading to enterprise repositories.

Cyber-Foraging

Cyber-Foraging

Cyber-foraging* is the leverage of external resource-rich surrogates to augment the capabilities of resource-limited devices

Two main forms of cyber-foraging

- Code/Computation Offload
  - Offload of expensive computation in order to extend battery life and increase computational capability
- Data Staging
  - Improve data transfers between mobile computers and the cloud by temporarily staging data in transit

* Satyanarayanan, Mahadev (2001). "Pervasive Computing: Vision and Challenges". IEEE Personal Communications (IEEE)

**006 Which brings us to a very key concept in technical cloudlets which is cyber-foraging.

So cyber-foraging is a concept that was coined in 2001 by one of our collaborators on campus, Mahadev Satyanarayanan, although he goes by Satya; and he referred to cyber-foraging as being to leverage external resources, resource-rich surrogates, in this case cloudlets, that would augment the capabilities of resource limited mobile devices.

When we talk about cyber-foraging we're really talking about two different types of cyber-foraging. We're talking about code or computation offload, which means that we're offloading expensive computation. Why? Because I'm

trying to extend battery life and I'm also trying to increase computational capability.

Or I'm talking about data staging; which is I'm trying to improve data transfers between mobile computings and the cloud by temporarily staging data in transit and potentially doing some processing on that data.

Cloudlet-Based Cyber-Foraging

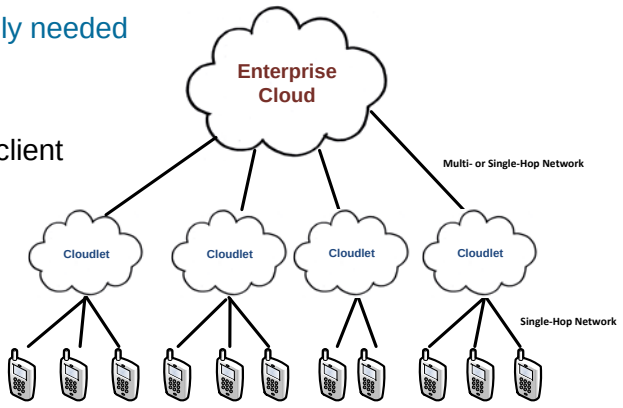
Cloudlet-Based Cyber-Foraging

Discoverable, virtual-machine based, forward-deployed servers located in single-hop proximity of mobile devices

- Can operate in disconnected mode
- Communication with the central core is only needed for provisioning

Applications are statically partitioned into a client and server

- Very thin client runs on mobile device
- Computation-intensive server runs on cloudlet



**007 So if I tie the two concepts together, I tie tactical cloudlets, which is what I explained earlier, and I tie it with cyber-foraging, what we're really promoting in our research is cloudlet-based cyber-foraging.

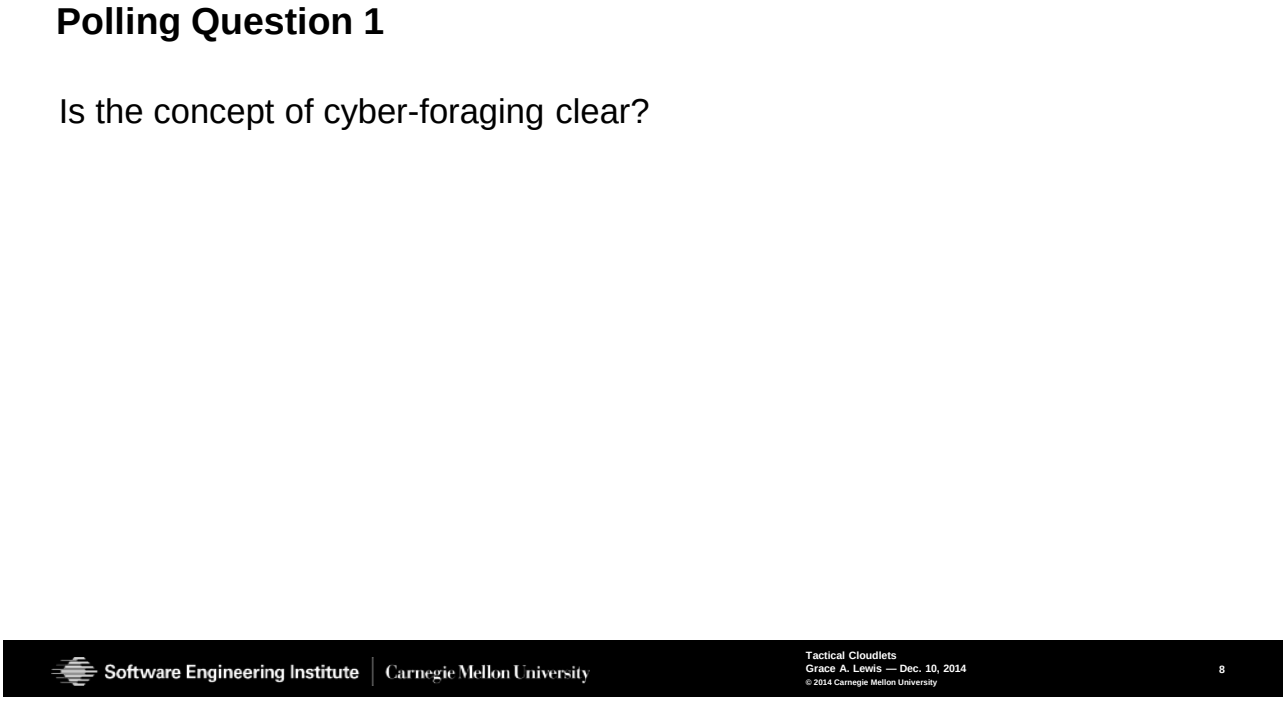
So in the case, going back to the diagram that I had earlier, you have

these cloudlets which are located in single hop proximity of the mobile devices-- so we'll talk about a single hop network-- and these cloudlets can be, or may not be, connected to the enterprise cloud but- and this can be a multi or a single hop connection.

So the idea is that they can operate in a disconnected mode; meaning that potentially I could have cloudlets pre-provisioned and I only need to communicate with that central core, just for the provisioning part.

In cloudlet-based cyber-foraging in order to take advantage of the types of applications that are used in the field, they're statically partitioned into a very thin client and a very- you could call it a very thick server, a computation intensive server that runs on the cloudlet.

Polling Question 1



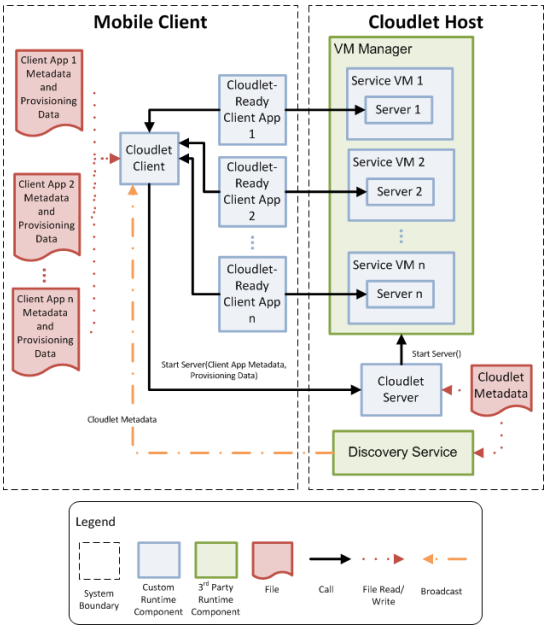
**008 Shane McGraw: Okay folks, going to lead to our first polling question today, where we want to make sure the audience is just understanding where we're at and kind of let you drive the flow of the presentation. So you'll see that as a popup on your screen now asking: Is the concept of cyber-foraging clear?

We'll give you about 10 or 15 seconds to vote on that; and based on that will drive the flow of the presentation.

Okay we'll close the poll now. We got 82% saying Yes Grace. So we are free to move on.

Grace Lewis: Wonderful.

Reference Architecture for Cloudlet-Based Cyber-Foraging



**009 So going back to the story that I'm going to be telling. We came up with what we call a reference architecture for cloudlet-based cyber-foraging. I will slowly try to go through some of the elements of this diagram.

So in a cyber-foraging system there are two main components that are noted by the dashed boxes.

You have the Mobile Client, which could be, for example, a smartphone; and a lot of our experimentation has been done with using smartphones as mobile clients.

And you have a Cloudlet Host. A cloudlet host as I said before, it can be a laptop, it can be a more powerful server. But the key is that

the mobile client and the cloudlet host are in single hop proximity.

What I mean by this is that they are connecting via Wi-Fi as opposed to 3G or 4G which is not only- which is not only more- it has higher latency but also it is known-- there are many studies that show that it consumes more battery.

So they're located in single hop proximity. They communicate using Wi-Fi.

So on the cloudlet host side there is an important part which is called a Discovery Service. So a Discovery Service, what that Discovery Service is doing, it's using a broadcast mechanism to basically tell the mobile devices that are around it: I'm a cloudlet, I'm a cloudlet, I'm a cloudlet. And it's transmitting some form of metadata; and we'll talk about the different options later.

So that's one main piece on the cloudlet host side.

On the mobile client side, the only piece of software that needs to run, in addition to the applications, is a Cloudlet Client.

So the cloudlet client on the mobile client side is able to detect that there are cloudlets in the area; and part of what the- part of what the Discovery Service is broadcasting is the information that allows the client-cloudlet client to communicate with a cloudlet server; which is the component that is on the right side.

And so basically the cloudlet client is in charge of establishing that initial handshake between the mobile client and the cloudlet host.

So what happens, once that initial handshake is established, is that the cloudlet client-- and I will talk about different forms of provisioning-- tells the cloudlet server what is the computation that it wants to run.

When the cloudlet server finds out what computation it needs to run, it basically gets- it basically starts up what we call a Service VM, a service virtual machine, that corresponds to the server portion of that application; it starts it as a VM inside the VM Manager and notifies the cloudlet client that it is ready to run; and therefore the cloudlet ready application can start interacting with that part of the server.

In general that's how cloudlet-based cyber-foraging works. So this is a reference architecture and what I'm going to show in the next slides are different ways in which we instantiated that architecture; and at the end I'm going to show you what-- based on some experimentation and some other work that we did-- what we ended up selecting as our cloudlet implementation.

Cloudlet-Based Cyber-Foraging: Operations

1. Cloudlet Discovery

2. Cloudlet Provisioning and Setup

3. Application Execution

**010 So for cyber-foraging to happen, three operations need to happen.

First there's a cloudlet discovery process. Then there's a cloudlet provisioning and setup process; and finally there's the application execution process.

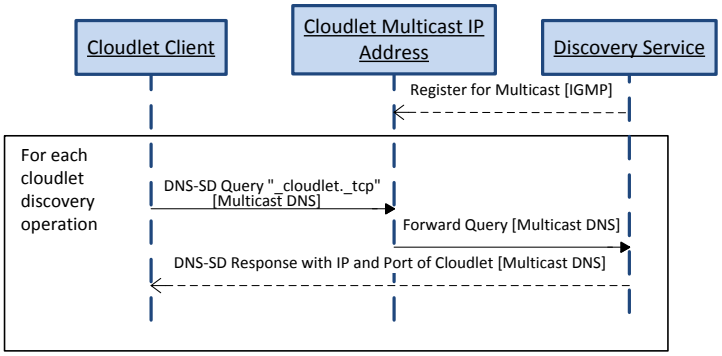
In the cloudlet discovery, as I said before, the mobile device discovers that there are cloudlets around it and selects a cloudlet.

And the second step is when we set up that cloudlet for execution, for executing the application that I want to run.

And finally the actual execution of the application.

Cloudlet Discovery

Cloudlet Discovery



Discovery Service implementation based on zeroconf

- Uses DNS Service Discovery (DNS-SD) along with Multicast DNS
- Multicast addresses are used to allow the client to request the service without knowing the IPs of the servers

Enables mobile devices to locate available cloudlets

**011 So is cloudlet discovery what do we do? So basically what we use as our discovery protocol, it's an implementation of something called Zero Conf, or Zero Configuration; and basically it's a-- it's based on multicast addresses in DNS.

So basically the cloudlet-- in advance what the Discovery Service does is it registers for multicast; and then when the cloudlet client is ready to discover a cloudlet, what it does is it says: Is there- are there any services out there that we've tagged with the tag cloudlet TCP? And based on that, the data is transmitted back to the cloudlet client.

So basically that is how- that is the mechanism that a mobile device uses or a mobile client uses for discovering or locating available cloudlets.

Cloudlet Provisioning

Configuring and deploying the Service VM that contains the server code on the cloudlet so that it is ready to use by the client running on the mobile device

Working prototypes for five different cloudlet provisioning mechanisms

- Runtime — provisioning from the mobile device
  - VM Synthesis
  - Application Virtualization
- Deployment time — cloudlets pre-provisioned based on mission needs
  - Cached VM
  - Cloudlet Push
- On-Demand — capabilities assembled at runtime
  - On-Demand VM Provisioning

**012 Now the second part, and probably the most crucial part of the process, is the Cloudlet Provisioning Process.

So basically cloudlet provisioning is the process by which we configure and deploy that Service VM that corresponds to the server portion of the application that wants to be executed.

We- as part of trying to select which was the best cloudlet provisioning mechanism, we instantiated five different versions of this reference architecture and we investigated five different cloudlet provisioning mechanisms.

We investigated two cloudlet provisioning mechanisms that are- in which the cloudlet is provisioned at runtime. One is called VM Synthesis;

the other is called Application Virtualization.

We investigated two which are based on- they're really done at deployment time, in which cloudlets are really pre-provisioned based on mission needs.

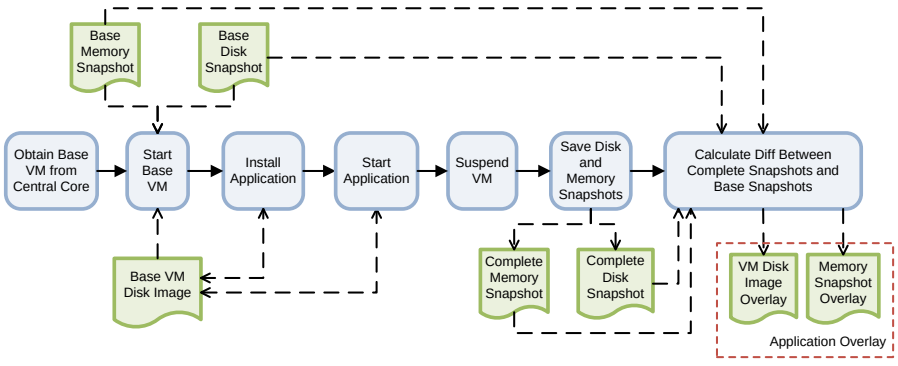
And then finally we tried a fifth approach, which is more or less a combination of our runtime and deployment time approach, where capabilities are assembled at runtime; and this resembles more of what happens in data centers nowadays and it's called On Demand Provisioning.

VM Synthesis 1

VM Synthesis 1

Cloudlet is provisioned by sending an application overlay from the mobile device to the cloudlet at runtime

Application overlays are created in advance for server portions of applications — they represent the diff between a baseline VM and that same VM with the application installed



**013 So as I said in one of my previous slides, the term Cyber-

Foraging was coined by one of our collaborators on campus, Satya; and when he envisioned this cloudlet concept, the way he envisioned provisioning these cloudlets was with something called VM Synthesis.

VM Synthesis is a very- it's a very novel and also complex and clever process for provisioning, where basically what is-- what happens with VM Synthesis is that the cloudlet is provisioned at runtime by sending something called an application overlay from the mobile device to the cloudlet.

Now application overlays have to be constructed in advance so that they can be stored on the mobile device and transmitted to the cloudlet at runtime.

So the diagram on this chart, basically what it explains is the VM Synthesis process.

So the key element behind VM Synthesis is something called a Base VM. A Base VM, what it basically is is a VM with- that has on it installed everything that an organization considers its baseline; its operating system, its security mechanisms, its-- anything that it considers its baseline is part of this Base VM.

And this Base VM is key to this concept because the Base VM has to be available to build the application overlays. But it also has to be available at runtime; and you'll see why.

So to construct a VM overlay what happens is the following. A Base VM is obtained from a central core. A VM, as you all- as all of you know, it's basically-- a VM manages a file.

So when I say it obtains the Base VM, basically what it obtains is two files from the central core. It obtains a base memory snapshot and it obtains a base disk snapshot. And so basically what happens at this point in time is that I start that Base VM; I start it- I start it inside my VM Manager.

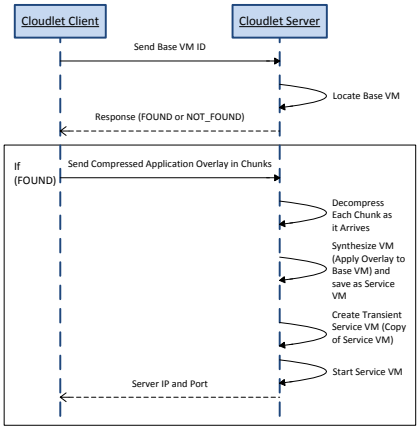
What I do after that is I install the application, I start the application, and then I suspend the application. When I suspend an application, what happens is that at that point in time the disk and memory snapshots are saved on disks. So what happens after that is I calculate the binary difference between the files after the installation process and the files before the installation process; and those two files, which are really a binary diff, constitute what is called the application overlay.

Now this happens at deploying time; and therefore for each- for each application that I want to run, I have to install the client for that application-- right?-- which is basically an app; and I have to install these two files which correspond to the application overlay.

VM Synthesis 2

VM Synthesis 2

The reverse of the application overlay creation process is done at runtime to create the Service VM



Applications	Payload Size (MB)	Application -Ready Time (s)	Client Energy (J)
FACE (Windows)	55	53.4	57.8
OBJECT (Linux)	332	175.7	333.3
SPEECH (Windows)	194	85.9	175.5
SPEECH (Linux)	147	99.0	172.5

Cloudlet Content	Exact Base VMs
Mobile Device Content	• Application Overlays • Client Apps + Metadata
Payload	Application Overlay
Advantages	Cloudlet can run any server code that can be installed on a Base VM
Constraints	Requires exact Base VM which limits distributions and patches

**014 So what happens at runtime is the reverse of that process. So what happens at runtime is that the cloudlet contains the Base VM. It receives the overlay and it applies the overlay to that Base VM. So what happens at the end is that I have this data which the VM was suspended in the previous slide.

So if you look at the sequence diagram on the left, what happens during VM synthesis, as I said before, is that the cloudlet client sends a Base VM ID to the cloudlet server. It locates that Base VM and it obviously returns whether it found an error or if it didn't find it.

If it did find it, then what the cloudlet client does is that it transmits the application overlay to the cloudlet

server. It decompresses it because it does the VM Synthesis process, which is what I talked about before, combining the Base VM with the overlay, and then it creates a Service VM and starts that Service VM, and it tells- now it tells the cloudlet client: I am ready and I am listening at this particular server- IP address and port.

Now this is- like I said before, it's the process by which cloudlets were envisioned; it was the initial process. But let's look a little bit at the data that we had when we ran some of these experiments.

So you will see that for all these provisioning mechanisms we executed experiments for four different applications. We executed with a face recognition application running on Windows, an optic recognition application running on Linux and a speech application, both a Windows version and a Linux version; because we also wanted to understand some of the differences there.

So as you can see from the payload size-- so the payload size-- and this is going to be the same for all the charts in all the other mechanisms-- is the size of the whatever-- in this case it's the overlay-- that is sent from the mobile client to the cloudlet at runtime.

Application Ready Time is measured as the time between-- a cloudlet client says: I need a cloudlet and I want it to execute this service; until

the cloudlet replies: I am ready for execution.

And cloudlet energy-- it's basically measured in joules-- is how much energy is spent by the mobile device during this process.

So as you can see, the payload, when we use VM Synthesis, is quite large. It ranges from 55 megabytes for a face recognition application all the way up to 332 megabytes for an object-recognition application.

Now if we tie this data back to our initial- our initial talk about the characteristics of edge environments, this is quite a large payload to send when you're talking about DIL environments, where you're talking about environments with intermittent connectivity where you're not really sure you're going to have the connectivity that you need.

Application Ready Time, compared to the other methods-- and I'll show you that data later-- is also quite large because the VM Synthesis process obviously takes time; and we've shown what many others have already proven, which is that payload size is really almost directly correlated to energy because data transmission consumes a lot of energy. So the larger the payload, the more energy is going to be consumed.

Now that is from a quantitative point of view. From a qualitative point of view, let's look at what has to- what

has to happen on each of the sides of the system.

So what has to be on the cloudlet, in addition to the cloudlet server and all that stuff? Well the exact Base VMs; meaning that I have to have the Base VMs from which the application overlays were built.

If you look at the mobile device side, I have to have the application overlays and I also have to have obviously the client apps and the metadata. The payload in this case is the application overlay.

Now advantages of VM Synthesis is basically that anything that I can install on a Base VM I can run. So that's a huge- a huge advantage because it gives you a lot of flexibility.

But the constraints are that you require the same exact Base VM; and this can become problematic when you're talking about distributions and patches because every time the Base VM changes, you would have to reconstruct all the overlays because it's only possible to reconstruct the overlay from the same exact Base

So that is- that is VM Synthesis and that is the first mechanism that we tried. And like I said before, I wanted to tell you a story.

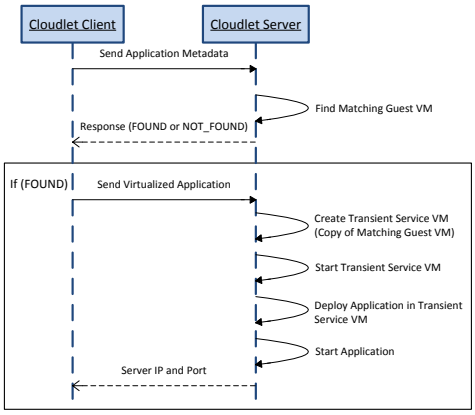
So yes, we were happy this worked. The concept that Satya had envisioned, it was perfectly feasible. But we were very concerned about payload size. So we started looking

into other approaches and other ways of doing this.

Application Virtualization

Application Virtualization

Cloudlet is provisioned by sending a virtualized application from the mobile device to the cloudlet at runtime



Applications	Payload Size (MB)	Application-Ready Time (s)	Client Energy (J)
FACE (Windows)	14	14.3	10.5
OBJECT (Linux)	29	21.9	24.5
SPEECH (Windows)	66	62.5	66.6
SPEECH (Linux)	68	38.3	54.2

Cloudlet Content	VM compatible with Server Code
Mobile Device Content	• Virtualized server code • Client Apps + Metadata
Payload	Virtualized Server Code
Advantages	Portability across OS distribution boundaries
Constraints	All server code dependencies have to be captured at packaging time

**015 So we looked into something called Application Virtualization. So basically what Application Virtualization is is you use tools-- like for example CD or Cameo-- you use them to basically create an application package; and inside that application package it contains everything that it needs to run.

So basically it's almost like OS virtualization where you're tricking an application into thinking that it's self-contained, that it's running independently.

So in Application Virtualization the package is no longer an application

overlay-- sorry, the payload is no longer an application overlay. The payload is a virtualized application; which is basically a package.

So the cloudlet-- this is another runtime provisioning mechanism-- so the cloudlet transmits the application-- sorry, the mobile device transmits the application overlay at runtime from the mobile client to the cloudlet.

So looking at the sequence diagram on the left. So the cloudlet client sends the application metadata to the cloudlet server. By application metadata it's basically saying: I'm a Linux-based application; I need this version of Linux or at least the OS family I'm talking about.

The cloudlet server finds a matching guest VM; it means it finds a VM that has an operating system that can run the virtualized application and it tells it if it found it or not.

If it's found, then at that point in time I send a virtualized application-- which once again is a package.

So in this case what the cloudlet server does is it starts up a copy of a matching guest VM. It deploys the application inside that VM and it basically starts the application and tells the cloudlet client that it's ready.

Now if we look at the data-- I'm looking- I'm talking about the upper right-- we'll see that the payload is much smaller- is much smaller than for VM synthesis. We're talking

about 14 megabytes as opposed to, for example, 55 from the last side for payload size for a Face application; and the largest in this case is Speech, both the Windows and the Linux version, with 66 and 68 megabytes. It's a lot smaller.

And that, like I said before, there's a direct correlation between payload size and client energy-- it decreases the amount of energy spent on the mobile device-- and application ready time is also diminished, not only because it's sending less data but also because really there is no VM synthesis process. All you're doing is really taking a packaged application and putting it inside a running VM- a running VM, yes.

So in this case what is the cloudlet content? The cloudlet content is that you have to have a VM that is compatible with the server code that I just sent over.

On a mobile device I need to have the virtualized server code, which in this case is an application package; and of course I have to have my client apps and my metadata; and the payload is that virtualized server code or package.

Now advantages are really portability across OS distribution boundaries because all that you need to know is I'm going to be running a Linux-based application or a Windows-based application; and then- and that gives you a lot of flexibility.

Now the constraints is that the tools that are available nowadays to build these application packages are not perfect. They're good but they're not perfect. Because what happens when you're creating one of these packages is that you have to run these tools; and what they try to do is they try to capture all possible dependencies of the application so that it can be- it can be self-contained in this package.

And there's a lot that you can miss; whether you're trying to look at the way the product was installed or the way the product is running, there's a lot that could be missed or there is a lot of-- there's a large margin for error because if you missed a dependency your virtualized application is not going to work.

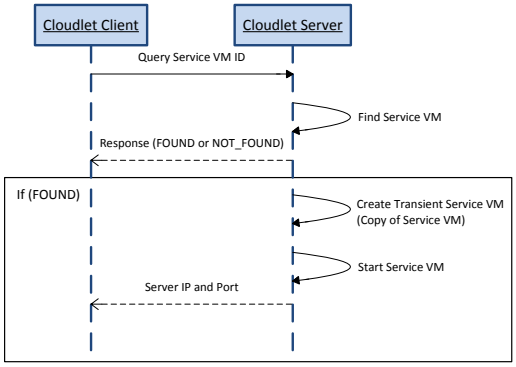
So on the one hand we were able to reduce payload size, application ready time and client energy; but we were introducing some error because there's a possibility that not all dependencies could be captured.

So we said: Okay, what if we just don't do this at runtime at all? What if we for a moment assume that we can pre-provision cloudlets? And one of the things I'm going to say at the end is that if you look at edge environments--

Cached VM

Cached VM

Cloudlet is pre-provisioned based on mission needs
Repository of VMs that contain capabilities
Each VM is treated as a service



Applications	Payload Size (MB)*	Application-Ready Time (s)	Client Energy (J)
FACE (Windows)	0.00	8.2	10.3
OBJECT (Linux)	0.00	11.6	13.5
SPEECH (Windows)	0.00	12.2	14.7
SPEECH (Linux)	0.00	12.2	14.9

* Size of payload is less than 1KB

Cloudlet Content	Service (VM) repository
Mobile Device Content	Client App + Metadata
Payload	Service ID
Advantages	Supports server code updates as long as service interface remains the same
Constraints	Cloudlet is provisioned with service VMs required by client apps (or has access to them)

**016 It's not too crazy to think about this- think about the fact that a cloudlet could be pre-provisioned at deployment time with all the VMs that I need based on mission need.

So if I'm going to do certain things-- I might need some mapping VMs, I might need some face recognition, some speech recognition, some situational awareness VMs.

So we looked into a technique that basically we called Cached VM. It's an extremely simple technique. But basically, like I said before, instead of trying to build- to transfer and to build these VMs at runtime, what I do is basically I have a cache of VMs; which is why we called it Cached VMs.

So I have a bunch of Service VMs in a service repository; and so what happens at runtime-- looking at the sequence diagram on the left-- is that I tell-- the cloudlet client tells the server: Do you have a service with this VM ID? So the VM IDs in our case were Face, Object, Speech-- whatever.

And so basically the cloudlet server finds a Service VM. It tells it: Yes I have that Service VM that you need. And what happens on the server then is that it creates a copy of that VM and it starts it up for the cloudlet client and it says: Here it's ready.

So the payload size in this case is-- it's really nothing because it's just a Service VM ID. Right? So you could say that the payload size is zero.

The application ready time is extremely fast because basically you already have a VM; and like I said before, we saved the suspended state. So starting it up is very quick; and obviously that reduces client energy of course because the payload is so small, as well as the application ready time.

In this case what you have to have on the cloudlet is that Service VM repository. What you have on that mobile device is-- in addition obviously to the cloudlet client-- is client app metadata; and the payload, as I said before, is simply a Service ID.

So what are the advantages? The advantages are-- well in addition to what I just said, you know, small payload and small consumption of energy-- it's the fact that I can update my Service VMs without any problem; which was the problem with the first technique with VM Synthesis. Because as long as a Service ID remains the same, I can still find the Service VM that I- the ID remains the same, I can update the service and nothing happens.

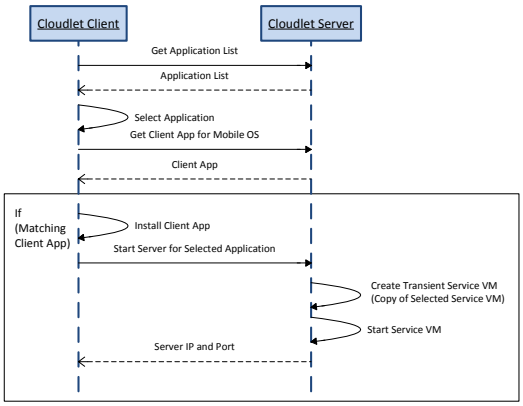
Now the constraint is that the cloudlet has to be pre-provisioned with these VMs that are required by an app. So it's-- I better do a good job of trying to figure out what should be on my cloudlet; and/or I could think about the possibility of the cloudlet being connected to the core to be able to obtain these Service VMs if I don't have them.

So again, that's the advantage-- lots of advantages. But it does have the constraint or the disadvantage that I talked about, which is that you do have to have these- have to anticipate what are the services that you're going to need.

Cloudlet Push

Cloudlet Push

Cloudlets are pre-provisioned and corresponding client apps are pushed to the mobile device at runtime



Applications	Payload Size (MB)*	Application-Ready Time (s)	Client Energy (J)
FACE (Windows)	0.0	7.9	13.8
OBJECT (Linux)	0.0	11.7	16.9
SPEECH (Windows)	0.0	12.8	18.2
SPEECH (Linux)	0.0	12.8	18.2

Cloudlet Content	Repository of Paired VMs (Server code) and Client Apps
Mobile Device Content	None
Payload	Client App and Metadata
Advantages	Supports most client mobile devices with distribution at runtime
Constraints	Cloudlet has a client app version that matches mobile client OS version

**017 Now if you're in the field, sometimes it might be difficult to have like a pre-provisioned mobile device with all the client applications that you need.

So what if we do something a little bit similar to Cached VM; but in addition to having a repository with Service VMs, what if we also have a repository with the client apps? So this is more or less like having an app store on the cloudlet; and we called this Cloudlet Push just because it was kind of pushing the application in this direction.

So in this case what happens at runtime-- I'm going to use the sequence diagram to explain what happens-- is that the cloudlet client would tell the cloudlet server--

instead of saying- instead of saying: Can you start the Service VM for me?; it's basically saying: Tell me what you have; what services do you have, what applications do you have?

And so the cloudlet server in this case returns a list; and once the-- and the cloudlet client can select from that list the application that it wants; like I said, similar to an app store.

It sends back the client app; and the client app is installed on the mobile device; and in the meantime what happens is very similar to what just happened in Cached VM, which is that the Service VM is started and basically the app is informed that it is ready to run.

So the payload size in this case is also very small, like in the previous case, because what it sends over-- in this case the payload is going from the cloudlet server to the mobile client-- is an APK or I mean just because we use Android; but it's an app file, it's small, especially because we were talking about having thin clients.

Application ready time is also considerably small because at the same time that it's starting the server on the server side, it's installing the client application; and that really doesn't take a lot of time because as I said before they're very small applications, which also leads to low energy consumption.

On the cloudlet, what do I have to have? Well I have to have a repository not only of the Service VMs but of their corresponding apps of course.

On a mobile device I really don't need anything other than the cloudlet client. So it's none.

The payload in this case is the client app and all its metadata; all that the client would need in order to install the application.

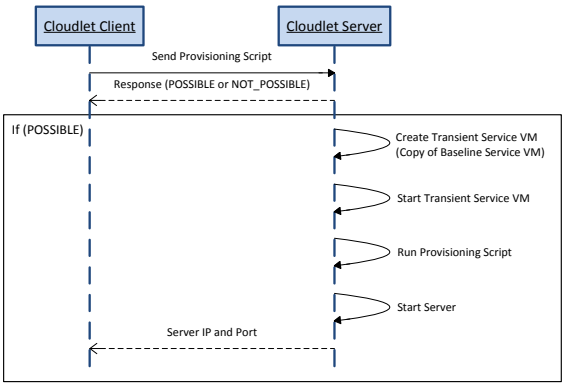
The advantage is that it really has distribution at runtime; so it can really support a lot of mobile devices and applications and mission needs.

But obviously the constraint is that if I'm going to have an app store or if I'm going to have to have paired Service VMs with their applications, I would have to have at least either knowledge or a variety of versions of the app that would match the mobile client that is being used.

On-Demand VM Provisioning

On-Demand VM Provisioning

Cloudlets are provisioned by assembling a server VM at runtime using a cloud provisioning tool, according to a provisioning script sent from the mobile device



Applications	Payload Size (MB)*	Application-Ready Time (s)	Client Energy (J)
FACE (Windows)	0.0	112.7	129.1
OBJECT (Linux)	0.0	211.0	244.0
SPEECH (Windows)	0.0	237.6	269.2
SPEECH (Linux)	0.0	94.1	109.3

Cloudlet Content	• VM provisioning software • Server code components
Mobile Device Content	• VM provisioning script • Client App + Metadata
Payload	VM Provisioning Script
Advantages	Service VM with server code can be assembled at runtime
Constraints	Cloudlet has all required server code components (or access to them)

**018 So we looked at provisioning mechanisms at runtime; we looked at provisioning mechanisms at deployment time; and we took a step back and we said: Okay what do real cloud data centers do; how do they provision VMs, how do they get a VM ready for execution?

And what we discovered, because it's common practice nowadays, is that what companies do, and organizations for example that are setting up their private clouds internally in their organizations, is that they use things called cloud provisioning tools.

So for example, Puppet; Puppet is an example of a cloud provisioning tool; Chef is another example of a cloud provisioning tool.

And so basically in the data centers, in the private clouds, what a client-- in this case a person or an organization-- would do is say: Hey I need a- I need a VM with these characteristics or-- and what the organization would have is a set of scripts that correspond to different versions of VMs that people would need.

And so what we did was something very similar. So on the right side- on the right side at the bottom right where I have the sequence diagram, in this case what is being sent from the cloudlet client to the cloudlet server is a provisioning script, which is basically a list of instructions saying: I want a VM and I want it to have these characteristics; so I want it to have these products in it.

And the cloudlet server in this case would respond: Yes I can put that VM together for you or not. And if it can basically what it does is that it starts up whatever it considers its baseline VM-- it's the same-- and it starts putting it together. It basically starts installing products in it. It basically runs the provisioning script; and when it's done it starts the server and it informs the cloudlet client that it is ready.

So it's very similar to the other- it's very similar to the other methods. But in this case what is happening is that it's being built at runtime; there's nothing prebuilt other than the provisioning script.

So in this case-- and you'll see some of the- some of the differences with the other mechanisms here-- the payload size is also very small because these scripts, we used- we used Puppet in our implementation; they're very small.

And but the application ready time is quite large; the application ready time is quite large because it takes time, it takes time to assemble these VMs because what the cloudlet server is doing is at runtime it's putting it together.

And this large application ready time leads to high client energy because even if the mobile device is just waiting, it's still consuming energy.

So you could-- we did other- we collected other data where we tried to subtract that time to try to make it just really the provisioning time. But in reality the application is just sitting there waiting for something to happen; and it is consuming energy.

So what does the cloudlet need to have in this case? Well it has to have VM provisioning software-- in our case, like I said, it was Puppet-- and it has to have all the server code components. What I mean is that it would have to have kind of like a repository of all the potential components that can go inside a virtual machine.

The mobile device needs to have the VM provisioning script and all the client apps and metadata, like the

previous ones. And as far as payload, what is sent from the client- from the client to the server at runtime is this VM provisioning script.

So the advantages is basically that it's really increased flexibility because pretty much as long as you have all the components you can build whatever Service VM you want.

But that- that's really its disadvantage as well because you would have to have access to all these components at runtime in order to put the VM together, whether these components already exist on the cloudlet itself or whether you're obtaining these components from other repositories, in which case you would need kind of like a connected cloudlet.

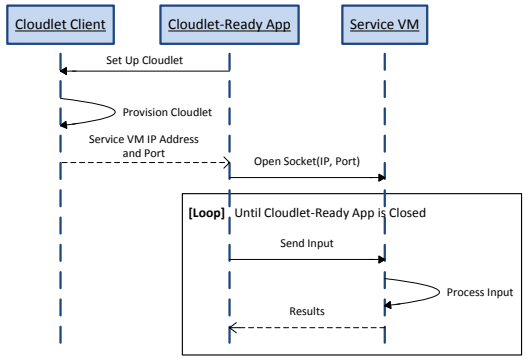
So that more or less is the summary of how we got to where we are now.

Application Execution

Application Execution

After receiving the Service VM IP address and port, the Cloudlet Client returns this information to the Cloudlet-Ready Client App

The Client-Ready App opens a socket to the Service VM IP address and port and executes in client/server mode until the app is closed



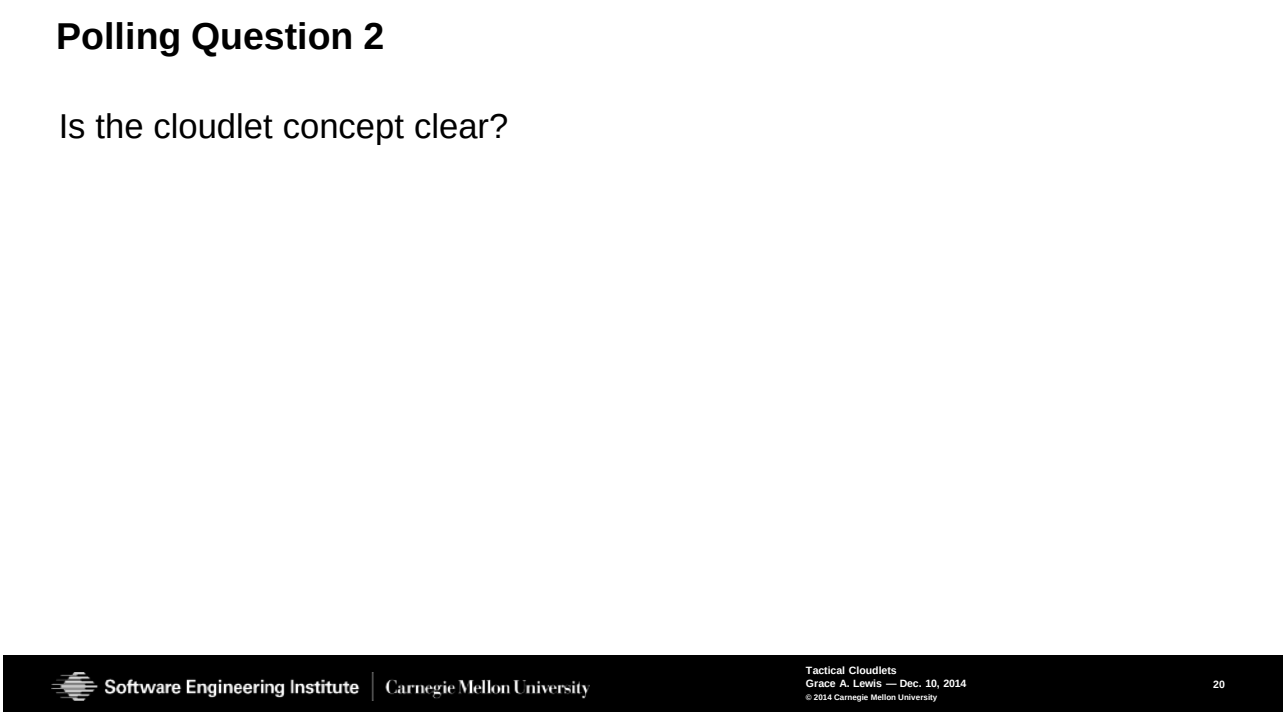
**019 By starting these different mechanisms.

So once the cloudlet is provisioning, all five mechanisms, regardless of which mechanism you work- you use, work exactly the same.

So after receiving the- after the client receives the Service VM, IP address and port, what the cloudlet client does is it returns this information to the cloudlet ready app and the cloudlet ready app basically opens a socket to the Service VM and starts interacting with it in client-server mode until basically the app is closed.

So the application-- like I said, it's very simple. They all function the same way. You just open a port and you start interacting with it in a pure client-server way.

Polling Question 2



**020 Shane McGraw: Okay this'll be just polling question two and we'd like to know-- and I'll pose that now to your screen-- Is the cloudlet concept clear?

And we'll give you about 15, 20 seconds for that. While we're waiting for that Grace, let's get to a question from Tim asking: How does cyber-foraging relate to mobile cloud computing?

Grace Lewis: Okay that's a good question. So mobile cloud computing is-- it's more of an overarching term which describes, as the name indicates, the intersection- kind of like the intersection or interaction between mobile computing and cloud computing.

So there are I would say three different flavors of mobile cloud computing. The first flavor of mobile cloud computing is simply when you have an app interacting with the cloud; like for example when you'd use Facebook and things like that. So you have a mobile app that is using the cloud to fulfill part of its functionality. Right? That would be the most basic form of mobile cloud computing.

The second flavor of it- of mobile cloud computing is when you're leveraging other mobile devices to kind of form your cloud.

So I need to execute an operation. I know that my mobile device either doesn't have all the data or doesn't have all the computing power to execute it. So I look to other mobile devices to see if maybe as a whole we can execute that computation. That's another form of mobile cloud computing.

Now cyber-foraging would be the third form of mobile cloud computing. So that's basically-- it's a form of mobile cloud computing.

Shane McGraw: Tim, thank you for that question.

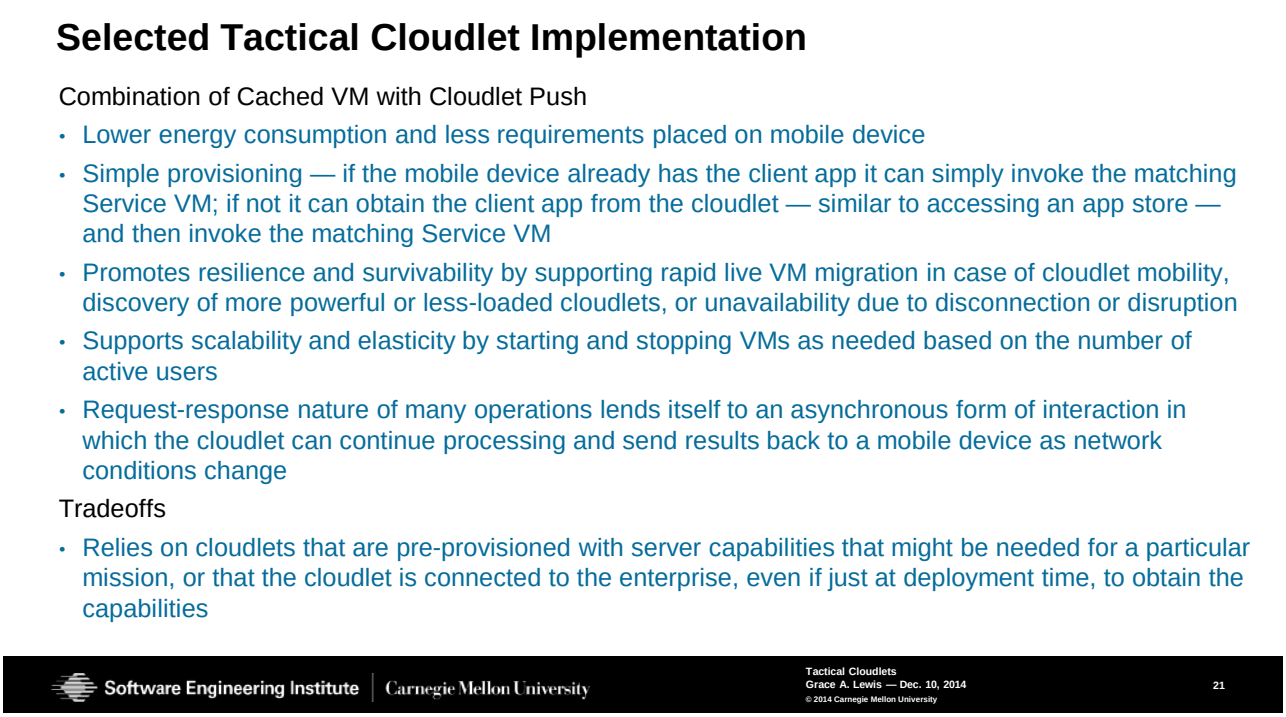
Okay back to our poll. We got about 93% are clear on the cloudlet concept. So we can move on.

Grace Lewis: We're getting better.

Shane McGraw: Yes.

Grace Lewis: Okay.

Selected Tactical Cloudlet Implementation



**021 So this is where we are now. So we did a lot of experiments. We built a lot of prototypes. We gathered a lot of data because we wanted to make sure that whatever we selected as our technical cloud implementation would be the best for the characteristic of- characteristics of edge environments that I mentioned before: the dynamic context, the intermittent connectivity, the lower computation power. And what we ended up doing was a combination of Cached VM and Cloudlet Push.

Now the data showed us definitely that there was lower energy consumption and less requirements

placed on the mobile device because if you combine Cached VM and Cloudlet Push, you have the freedom of really not having anything at all on the mobile device; or maybe already having some apps on it.

It's also- we also consider it a very- a simple form of provisioning in the sense that when you use VM Synthesis you have the-- I don't want to say the trouble-- but you do have to create the VM overlays. When you have application virtualization, you do have to create all these package applications.

So we thought of- combining Cached VM and Cloudlet Push would give us very simple provisioning because first of all, constructing Service VMs is not- it's not hard at all, you basically take a VM and you install something in it and you have a Service VM.

And from the mobile client perspective at runtime either I don't have the application, in which case I use Cloudlet Push, or I do have the application, in which I'm basically using Cached VM.

Also we opted for- or we agreed that doing a VM-based approach would be much better for the types of environments that we were operating in because it does promote resilience and survivability; and what I mean by this is because-- for example, live VM migration is an out of the box capability supported by many of the virtual machine managers out there.

You can imagine a situation where you can migrate a VM, a running VM, from one cloudlet to another; and that will allow me to continue operations, whether the reason why I have to migrate is just mobility-- I'm moving far away from this cloudlet and I'm getting closer to this cloudlet-- or this cloudlet is really loaded, so I'm going to move some of my load over to another cloudlet; or it could be something as simple as I'm working here, a vehicle came up-came to pick me up, I'm going to move to the cloudlet that's on the vehicle, so I'm just going to manually move all my computation from this cloudlet to this cloudlet.

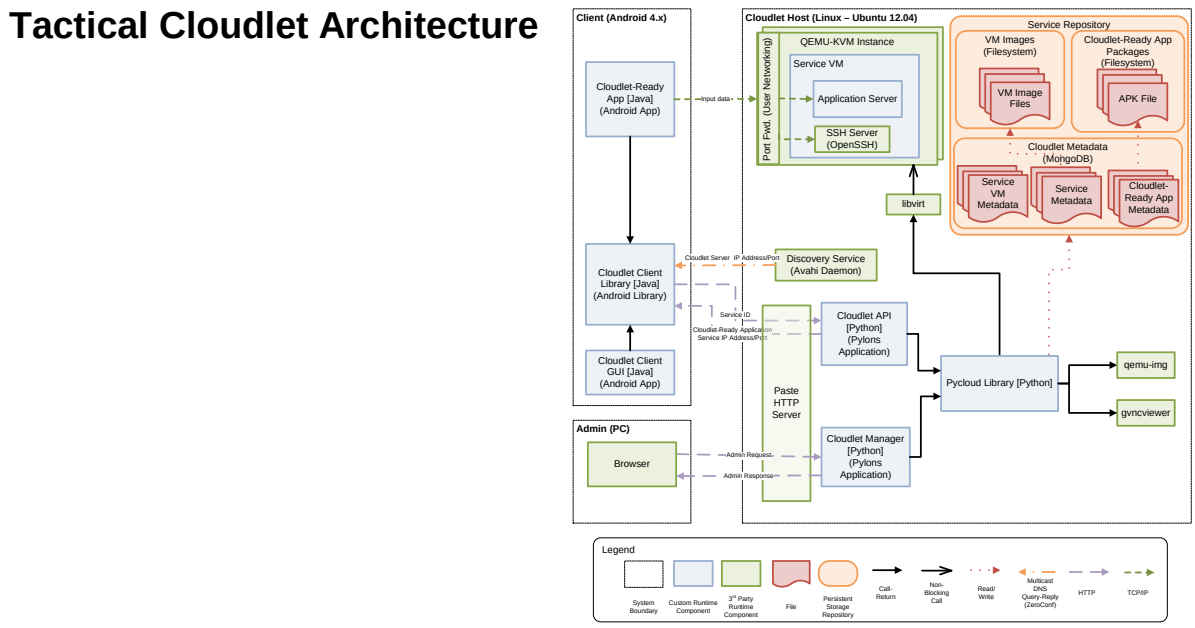
Obviously it also supports scalability; and this is just basically what is done in the cloud data centers because we can start and stop VMs based on the needs that we have.

Also the request response, having a very thin client and a very thick server that operate in a request/response manner, they seem to be much better for DIL environments because you're basically saying: Do this computation for me; here's your computation. Instead of maintaining state over time.

Now there are tradeoffs of course-- and this is something that we recognize and we accept it as part of our implementation-- is that it relies on cloudlets that are pre-provisioned.

We assume that the people that are putting these cloudlets together know what capabilities are going to be necessary for this particular mission or that maybe even if it's just at deployment time or at intermittent periods we're going to be connected to a core or a data center to obtain some of these capabilities.

Tactical Cloudlet Architecture



**022 So this diagram on this slide is showing the tactical cloudlet architecture; and now it's an instantiation of some of the products that we selected as part of our implementation.

So as I've hinted to on many of my slides and during my talk, we are using Android as our mobile platform; and as far-- and as our- on the

cloudlet host we're using a Linux-based cloudlet host; and as our virtual machine manager we're using KVM, simply because it's very simple to use and it really has a lot of the capabilities that we need. It also has a very small footprint which is also another characteristic of edge environments and some of the tactical environments.

And so some of the products are there. Like I said before, for discovery we're using simply an Avahi daemon, which is an implementation of Zero Conf.

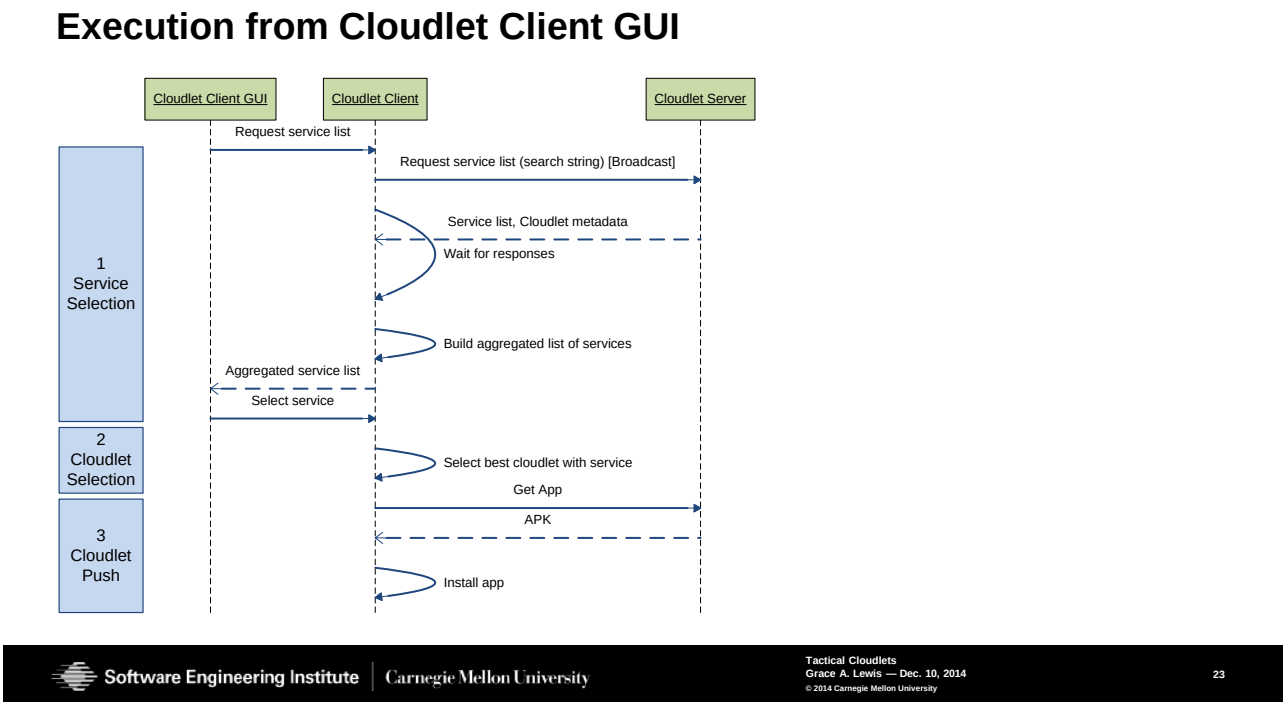
Now what is a little bit different from the reference architecture is that because we selected to use a pre-provisioning mechanism, we do need some type of management console; which again is another concept that is very common in cloud computing environments, which is why this talk is called Moving Cloud Computing to the Edge.

So we have a simple management console that also runs on the cloudlet server and allows you to install Service VMs, to start Service VMs, to stop Service VMs, and to just see what you have on a cloudlet; and that is the Cloudlet Manager that is towards the bottom.

And the Cloudlet API is the API that allows us to do a lot of these operations that I talked about before that enabled both Cached VM and Cloudlet Push.

So basically that is more or less the implementation that we relied on. As you will see, we also have a VM- a service repository. The service repository has the VM images that correspond, in this case, to Service VMs; and it also has the cloudlet ready app packages as APK files because we're using Android; and we just used MongoDB simply to sort that information because it's also very simple to use.

Execution from Cloudlet Client GUI



**023 So how does this implementation in the end work?

So you'll see on the left side of this slide that we have like three different pieces. We have Service Selection, Cloudlet Selection and Cloudlet Push.

So as part of our implementation-- I haven't talked about this in the previous slides-- we also accept the fact that I could be in proximity of multiple cloudlets; so there has to be some kind of cloudlet selection process.

So so far all the data that we showed, and also a lot of the explanations that we showed, assumed that there was only one cloudlet around. But what if there's more than one cloudlet? And this particular- our baseline implementation accepts that possibility.

So in this case the cloudlets-- if I'm going to use from- if I'm going to execute from the cloudlet client GUI, it means that I don't have the application, which means I have to go to the app store.

So in this case I tell the cloudlet client: What services do you have? And the cloudlet server responds with a list of services that it has from all the available cloudlets. I build an aggregated list of services and in this case from that aggregated list of services I select the service that I want. The cloudlet client selects the best cloudlet with the service.

Now one of the things that I mentioned at the beginning is that one of our goals from a R&D perspective is to have a very, very flexible architecture because we want to be able to experiment with a lot of things.

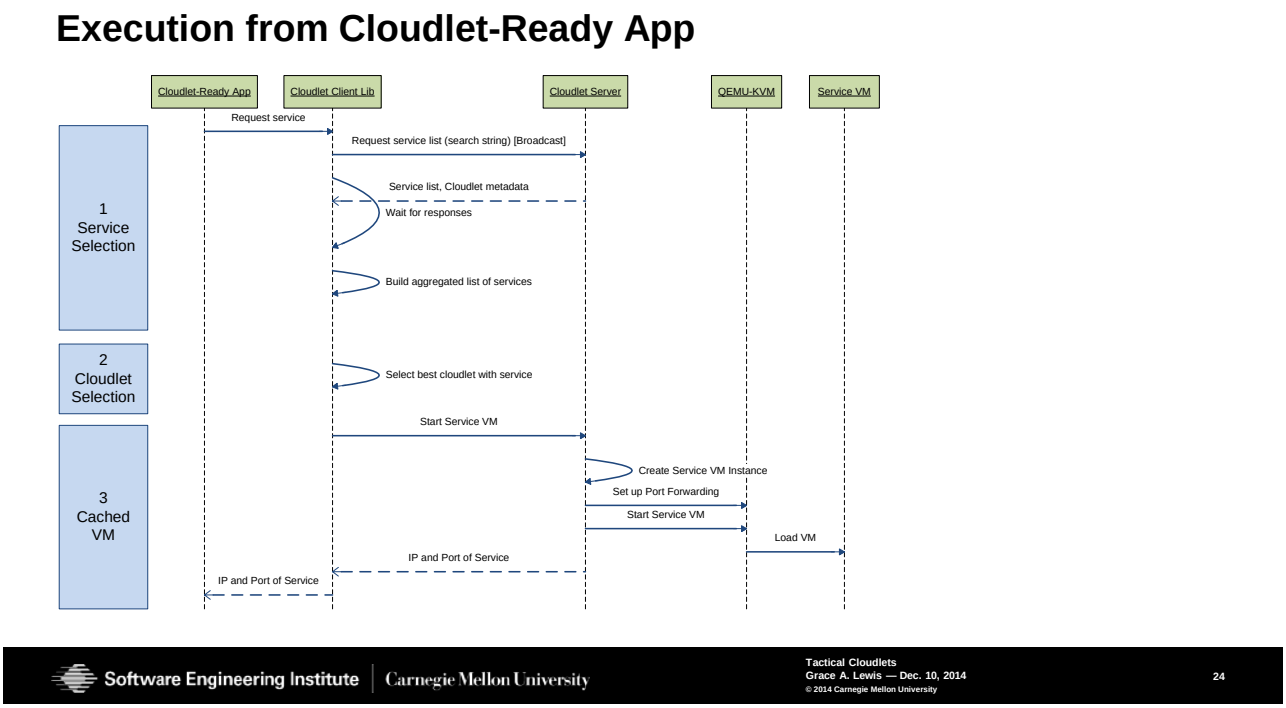
So in our architecture you can plug in what we called a Cloudlet Selection Algorithm. The algorithm that we're using at this point is a very simple algorithm. It just selects the cloudlet that is less loaded.

But you can imagine replacing that algorithm with any other type of algorithm; the cloudlet that is more powerful, the cloudlet that is closer-- there are many algorithms that you can replace it with it.

In our case we just-- it's a very simple algorithm that chooses the less loaded cloudlet; and it gets the application from this cloudlet and it installs it. So basically this is kind of like the Cloudlet Push part of the equation.

Now what if I already have the application?

Execution from Cloudlet-Ready App

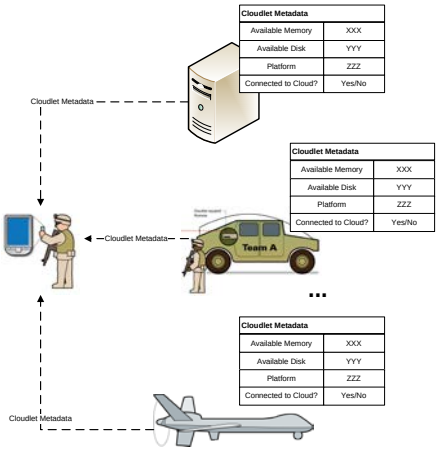


**024 Well if I already have the application, what I'm doing is more or less- more or less Cached VM.

So in this case I am requesting a particular service. It's also building the list; it is selecting the best cloudlet that has that service and now it's simply starting the Service VM and returning.

So this is what I mean by that we combined Cached VM with Cloudlet Push. So you can either have the application or you don't have the application.

Current and Future Work



Standard packaging of Service VMs

- Installed from the cloudlet manager, enterprise Service VM repository, thumb drive, or the mobile device connected via USB to the cloudlet
- Capabilities to improve mobile systems survivability
- Optimal cloudlet selection
- Cloudlet handoff (live migration) — manual and automatic based on load and other attributes
- Support for data-reliant systems running on cloudlets disconnected from the enterprise

Focus for FY15 will be Trusted Identities in Disconnected Environments

**025 So where are we now? So that was the past. I'm getting into a little bit of the present; and now I'm going to go a little bit into the future.

So things that we have done based on that cloudlet implementation.

So we've come up with what we called the standard packaging of Service VMs. I'm not going to call it a standard; but we came up with a standardized way of representing Service VMs.

Why is that the case? Because we want to- we want people that are using cloudlets to be able to load Service VMs either using the Cloudlet Manager or maybe there is an enterprise level Service VM repository or maybe I have it on a thumb drive

or maybe it could be the case that I have it on my mobile device and I just want to connect my mobile device to the cloudlet.

But we want to be able to- be able to obtain these capabilities from multiple mechanisms; and in that case you need some kind of standard- a standardized way of packaging.

We've also added capabilities to improve mobile systems, what we call survivability. So we added- like I said before, we added an optimal cloudlet selection algorithm.

We've implemented a cloudlet handoff, which is live migration.

We are currently able to do it manually; which means that we are able to tell it: I want you to- I want you to move this computation from this cloudlet to this cloudlet.

And at the moment we're working on what we call automatic migration; which is being able to move that VM based on other characteristics, like I said before, such as potential disconnection or mobility or excessive load or things like that.

We are also-- this year we're hoping to be able to work on support for data-reliant systems.

So so far a lot of the applications that I've described are very what I would call self-contained, in the sense that if I want to- if I want to use face

recognition-- I can have my face data based on the cloudlet-- I don't need to reach back to the enterprise.

But what if I have a scenario in which I do need to reach back to the enterprise, whether I have to report on data that I'm seeing or I want to be able to consolidate with data from other cloudlets around the world?

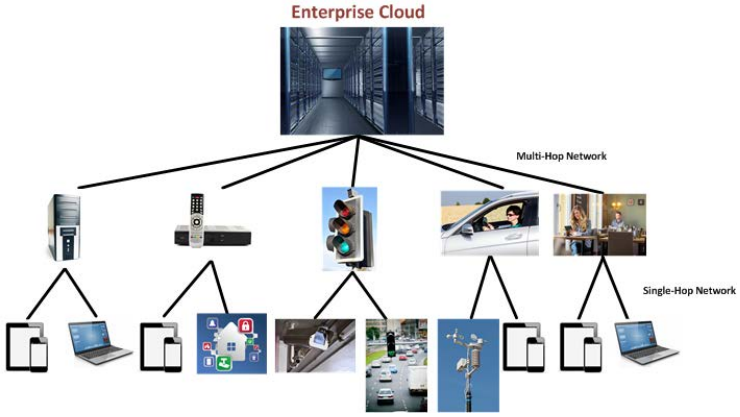
So what we're trying to do is add these capabilities for- that are specific data-reliant applications. How do we support data-reliant applications in a situation where there might not be connectivity back to the enterprise core but we want to keep on functioning even if we're disconnected from the enterprise?

And specifically for FY 15, for Fiscal Year 15, we're working on Trusted Identities in Disconnected Environments.

So if you look at the way we're doing Discovery-- you know, what a mobile device does is basically say: Oh are there cloudlets around there? Well as a cloudlet, I want to make sure that whoever is asking for me is really a friendly person; and from a mobile device perspective I also want to know if I see a cloudlet that's saying, "I'm a cloudlet, I'm a cloudlet" I want to make sure it's also a friendly cloudlet.

So that is- from a research perspective, that's our focus for this particular year, for Fiscal Year 15.

Cloudlets: Beyond Tactical Environments



Goals

- Bring the cloud closer to the user
- Support rich sensing and interaction capabilities of mobile devices seamlessly fused with compute-intensive and data-intensive processing

**026 Now let's look at cloudlets beyond tactical environments. So so far I've explained cloudlets from the tactical perspective, from the perspective of our stakeholders here at the SEI.

But let's look beyond tactical environments because-- and it's going to be on my last slide-- me and also our research group, we strongly believe that cyber-foraging is going to become almost like a built-in capability in mobile applications; and we'll see why.

So think of cloudlets outside of tactical environments. Think of having cloudlets inside your home; you know, that your cloudlet-- you can work standalone on your applications, on your pads and on

whatever when you're away from home. But once you get home your mobile device is able to detect that you are within your- within proximity of your home cloudlet and now it executes all that expensive computation and stuff on your home cloudlet so that you're not draining the battery on your mobile device.

Another home scenario. So think of your cable company. Those of you that have cable service have a cable box in your home. What if we added some more computing power to that cable box?

And for example, you can have a situation where your cable company forms some form of alliance with a video streaming company.

So in that case you could have your cloudlet cache a lot of your movies, a lot of your games, a lot of the things that you play; and what you're gaining with that is really an improved user experience because now when you're watching movies on your mobile device or when you're trying to play online games on your mobile device you're not going to the cloud for every interaction; you're going to your cloudlet which is right there in your house. So it really improves the mobile experience because you're really decreasing latency.

And another way in which you could use these cable boxes: Obviously for all the home automation and home security and things like that.

Another scenario-- and this is something that we actually had conversations about it with potential stakeholders-- is to use it for traffic control or for connected vehicles or things like that.

So imagine a situation where you have cloudlets in traffic lights and the traffic lights are running these cloudlets because they have the power to run them-- there are servers attached to the cloudlets-- and you're able to use these cloudlets to, for example, to do- to control traffic, to inform of accidents and to support a lot of the vehicle to vehicle communication that is being promoted nowadays.

Go in a little bit to the right. The next scenario, you see a person in a car. So imagine this is a field researcher. What is on the bottom is a weather monitor; and for example you could have sensors in the field collecting data and sending them to a cloudlet that is in this researcher's vehicle. The vehicle stopped to do some research and the cloudlet is running in the vehicle.

And also that researcher is also doing a lot of work on the mobile device. It's trying to execute some data, it's capturing some samples, wants to process those samples.

You could also think of this as a medic. Imagine a medic- a doctor in a type of- like an impoverished environment-- let's put it that way-- trying to run some of these very

expensive diagnoses, not being able to do it on the mobile device but being able to take advantage of the cloudlet that is running in that doctor's vehicle to do some of that. That's another potential scenario for cloudlets.

And finally there is the- well there's the Starbuck's scenario; which is just sitting in a coffee shop.

So what if- what if cloudlets became so pervasive that they're available everywhere? You can go to your favorite coffee shop and sit in your coffee shop; and your mobile device is capable of discovering that there is a nearby cloudlet and it's going to try to play your favorite game on that cloudlet and not on your mobile device, so you don't use up all your battery.

Or maybe, you know, all the runtime provisioning mechanisms that we talked about at the beginning are becoming better and maybe I can just say: Look I was working on- I was working on this; can you please- can I please kind of send it to you so I can work on the server in the meantime and then get it back?

So there are lots and lots of applications beyond tactical environments that I also might want to make you aware of; because in the end the goal of what we're trying to do is really to support this rich sensing and these rich interaction capabilities with mobile devices so that it's seamless and- it's seamless

and I'm executing either computation intensive or data intensive applications.

Mobile Device Trends

Mobile Device Trends

Smartphones and tablets have become for many the preferred way of interacting with the Internet, social media and the enterprise

- Number of smartphones has passed the number of laptops and desktops[1]
- Growth rate of e-readers and tablets is higher than smartphones, and if continued will reach high numbers soon[2][6]
- Smartphones and tablets are becoming the main computing device for many users[3][6]
- Not uncommon for there to be multiple mobile devices per user and household[4][5]

Organizations are pushing out more and more content and functionality to mobile users

**027 And where I am going with this? Where I'm going with this is-- and this is why I believe that cyber-foraging is going to become a main component-- is that there are so many studies, and you'll see the references at the end of the presentation, that are showing just the pervasiveness of smartphones and tablets.

The number of smartphones has already surpassed the numbers of laptops and desktops that are being bought nowadays. The growth rate of eReaders and tablets is growing- it's growing-- it's not higher than

anything yet but if the rate continues, it's going to be a matter of a very few years.

Smartphones and tablets are becoming the main computing device for many users; you know, people want to be able to do their computation on their couch and they don't want to have their laptop on, they just want to use their tablet.

And also it's not uncommon for there to be multiple mobile devices per user and per household. I know that my household is guilty of that.

And what is this- what is this doing? It's really driving organizations to push out more and more and more and more content and functionality to mobile users.

Therefore …

Therefore …

Not unreasonable for users to expect the performance and capabilities of mobile devices to be equal to laptops and desktops

However …

- Mobile devices will always lag behind their PC counterparts due to size and battery limitations
- Large and variable end-to-end latency between mobile device and cloud, and the possibility of disruptions, have a negative effect on user experience
- Will only get worse with the amount of network traffic generated by IoT

**028 So what's-- so what is that causing? Well it's-- because your tablets and your ereaders are becoming your main computational device, you're expecting to be able to do- what you do on a desktop to be able to do it on your smartphone-- right?-- or your tablet.

However we really have to keep in mind what I said before. Mobile devices are always going to lag behind their PC counterparts due to size and also due to battery limitations.

There is lots of research going on in batteries but it's not advancing at the same pace as other- as research in like hardware and software.

Also the more and more people that use mobile devices, the more and more data that is getting to the internet and the more congested our networks are becoming; and so relying on the cloud to provide a rich user experience is-- it's going to become impossible because you just- you're just not going to be able to support all these mobile devices and all this data and it's only going to get worse with the concept of internet of things.

So if now our appliances start transmitting data and there are sensors transmitting data and our mobile devices transit data, it's just a lot of data; which is why I strongly believe that something like cloudlets or some type of intermediate between the cloud and the mobile device is going to become a reality.

Food for Thought

With increasing number of mobile devices and users, increased network traffic cause by IoT, and increasing complexity of user experience, cyber-foraging will become a standard feature of mobile applications

Requires mobile applications and infrastructures to be architected and designed to adapt to a changing environment in which resources with greater computing power are discovered and used opportunistically

While the benefits in terms of mobile user experience and new business opportunities are huge, it requires a different paradigm in mobile software engineering

**029 And that's why I think it's going to become a standard feature of many mobile applications just because it's the environment that is pushing us this way.

Now from a software engineering perspective what is going to happen is that if this becomes a reality then we're going to have to build our applications in that way. We have to- we're going to have to build our applications to be able to sense for these cloudlets around us, these other resources on which we can offload applications.

And the benefits are huge; the benefits are huge in terms of business opportunities. Just think about some of the examples I talked about.

The benefits are huge in mobile user experience. But it's going to require a shift in the way we do software engineering because now we have to think about not only the resources that we have but the resources that we could have.

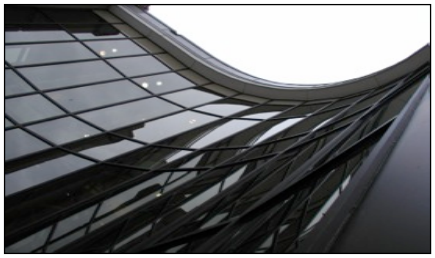
Contact Information

Contact Information

Grace A. Lewis
Advanced Mobile Systems (AMS) Initiative

Software Engineering Institute
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Phone: +1 412-268-5851
Email: glewis@sei.cmu.edu
WWW: http://www.sei.cmu.edu/staff/glewis/



**030 So with that, I think I would like to end.

Shane McGraw: Okay. Terrific presentation Grace; and before we get into Q&A just a reminder that a survey will pop up at the end of the presentation. We request that you fill that out as your feedback is always greatly appreciated.

So let's get into some questions for Grace.

Grace Lewis: Sure.

Shane McGraw: One from Bart asking: What about Denial of Service attacks? It seems simple to me to keep a cloudlet busy such that it cannot provide services to other clients.

Grace Lewis: No absolutely, absolutely. So the environments in which we operate in, like I said, are tactical environments. The advantage that we have-- and I'm not saying that Denial of Service is not possible-- but the advantage that we have is that it's more of a bounded environment; meaning that we know- we know-- so if we- for example, if we deploy a cloudlet per team, we know how big the team is. Right? So we have a bounded environment.

But if we do move towards- if we do move towards the more unbounded environment-- so going- so if I go back to my slide that had all the different potential scenarios, if I go to the coffee shop scenario where anybody could use a cloudlet, then it does.

One of the- I think one of the big challenges is going to be security in general-- not just denial of service but security in general-- because there are environments such as the one we operate in where we can have a lot of control over who uses a cloudlet; but there are others that- where you don't have that control.

Shane McGraw: Okay. Next one from Joan asking: By the Edge-- so do you mean the same thing as in edge computing?

Grace Lewis: Oh yes, so yes I should've clarified that at the beginning. Yes when I say "the edge" I mean basically the same as in Edge computing.

I'm sure all of you are aware Edge computing is a term that was coined by a company called Akamai many years ago where what they wanted to do was to push web content close to where the users were accessing it. So basically what they would do for an organization, if you hired Akamai, was that it would deploy copies of your website in multiple servers around the world so that people using it from different locations would get a better experience.

This is exactly what I mean by Edge computing; putting-- and actually I would argue that it's more than Edge computing because I'm not only taking content to the edge but I'm also taking computation and data and all that goes with it.

Shane McGraw: Okay. So a number of questions asking just about slides and recording. So the slides are available now in the Files tab within your console. You can download them now. The archive will be ready by tomorrow or the next day. We'll send out an email with a location for that.

So the last question in the queue-- so if you have any other questions feel free to type them in now-- but from Lawrence asking: What would it take to deploy a cyber-foraging system in the field?

Grace Lewis: Okay. So if you- if you believe that the tactical cloudlet implementation that we have is the way to go-- so what you need- what you would need is just a laptop if-- it really depends on how many users you're going to support.

But let's say you take a robust laptop into the field. You make sure that the laptop has the cloudlet software on it. What I mean by cloudlet software is kind of like the API part, the cloudlet server part of it, and also the cloudlet manager that I talked about. So you have all that deployed on your cloudlet.

And then you have mobile devices that are running the cloudlet client server; and that's really all you need because when you package all your- when you package all your cloudlet with all the Service VMs that you need, you're also packaging all the applications that go with it. So that's all you need. It's actually quite simple.

Shane McGraw: Okay. Terrific. Folks that's all the questions we have. So we'll wait about just another 30 seconds. Feel free to type them in. We're about 2:27 and we need to be done by 2:30. So we'll just wait to see if anybody else has any questions.

Just while we wait for that. You had mentioned security. Are you aware of any other security work going on at the SEI in mobile computing? Is that-- I know that's not in your expertise area.

Grace Lewis: Yes it's not, no, but there are; I mean, there are groups- there are groups within SEI, within CERT, that are trying to analyze vulnerabilities that are specific to- for example to Android and to iOS.

There's also other groups working on security from the perspective of-- so BYOD, Bring Your Own Device is a big trend, you know, where people want to be able to use their work devices- sorry, their home devices at work as well. So that obviously opens a lot of security vulnerabilities and breaches; and I know that there is other work at the SEI that is exploring that part of it.

Shane McGraw: Okay. Terrific. Folks, that's going to wrap it up. I would like to invite you to our next webinar. It will be on January 15th and it will be a Software Architecture virtual event with SEI Fellow Linda Northrop and Rick Kazman; and we'll send out an invite to everybody.

Thanks again for everyone- attending everyone. Grace, thanks again for the great presentation; and everyone have a great day.