A Reference Architecture for Mobile Code Offload in Hostile Environments

Grace A. Lewis (glewis@sei.cmu.edu)
Soumya Simanta (ssimanta@sei.cmu.edu)
Ed Morris (ejm@sei.cmu.edu)

Carnegie Mellon University Software Engineering Institute

Kiryong Ha (krha@cs.cmu.edu)
Mahadev Satyanarayanan(satya@cs.cmu.edu)

Carnegie Mellon University School of Computer Science

Motivation

First responders and others operating in crisis and hostile environments increasingly make use of handheld devices to help with compute-intensive tasks such as speech and image recognition, natural language processing, decision-making and mission planning

Challenges for mobile devices in hostile environments

- Mobile devices offer less computational power than conventional desktop or server computers
- Computation-intensive tasks take a heavy toll on battery power
- Networks in hostile environments are often unreliable and bandwidth is limited and inconsistent

Cyber-Foraging

*Cyber-foragin*g is the leverage of external resources to augment the capabilities of resource-limited devices

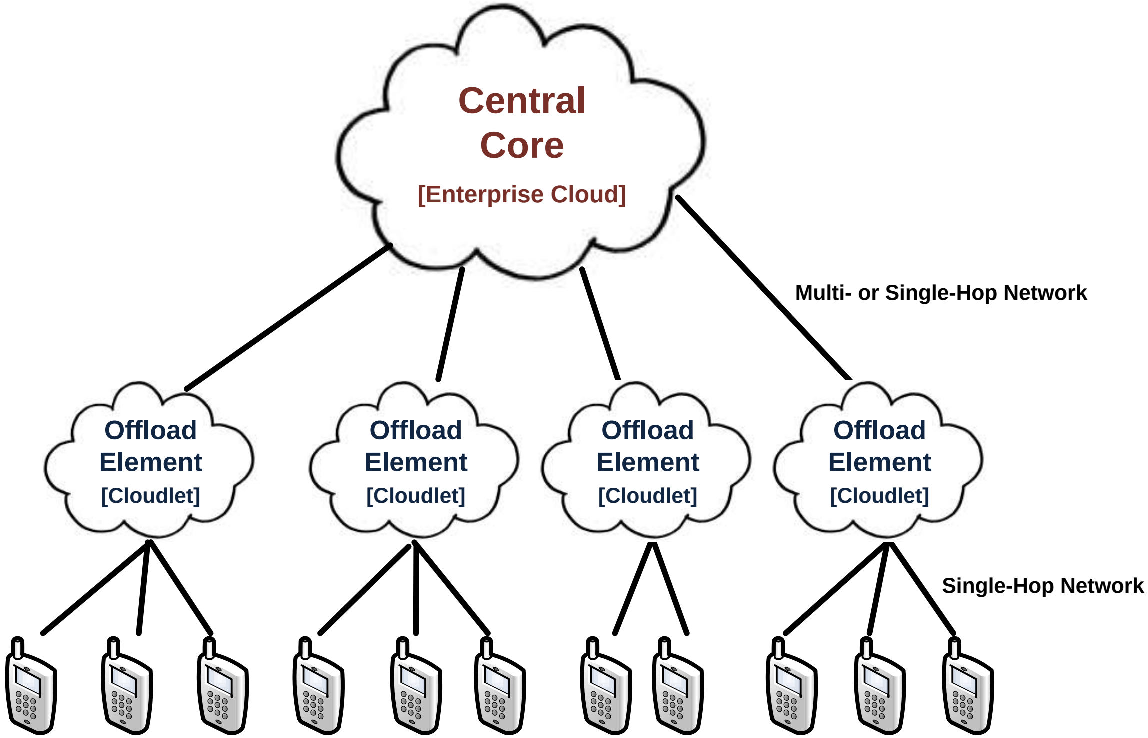
One form of cyber-foraging is *code offload* from mobile devices to the cloud to conserve battery power, increase computational capability, or to provide access to data resources

Most cyber-foraging solutions rely on:

- conventional Internet for connectivity to the cloud
- strategies that tightly couple applications running on handheld devices to the servers on which computation is offloaded

These solutions are not appropriate for hostile environments because they do not address the challenge of unreliable networks and dynamic environments

Code Offload in Hostile Environments*

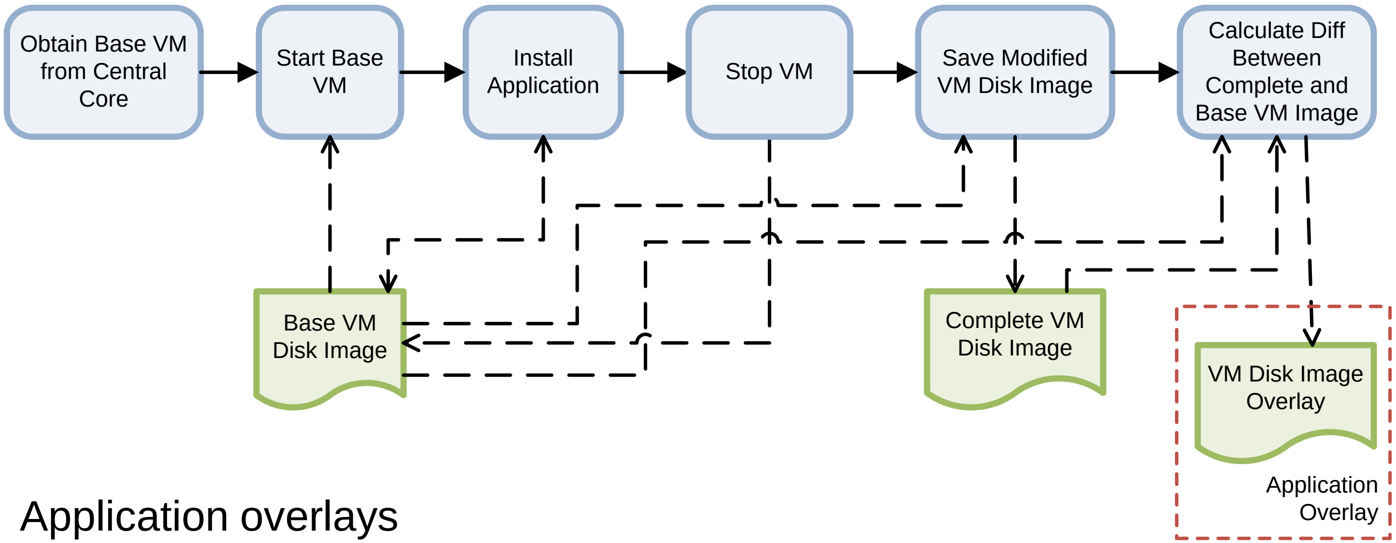


Cloudlets as Offload Elements

- Discoverable, generic, stateless servers located in single-hop proximity of mobile devices
- Run a separate virtual machine (VM) for each offloaded application
- Enhance processing capacity and conserve battery power while at the same time providing ease of deployment in the field
- Communication with the central core is only needed for provisioning

* K. Ha, G. Lewis, S. Simanta, S and M. Satyanarayanan. Code Offload in Hostile Environments. Carnegie Mellon University, CMU-CS-11-146, 2011. http://reports-archive.adm.cs.cmu.edu/anon/anon/2011/CMU-CS-11-146.pdf

VM Synthesis as a Strategy for Code Offload



Application overlays

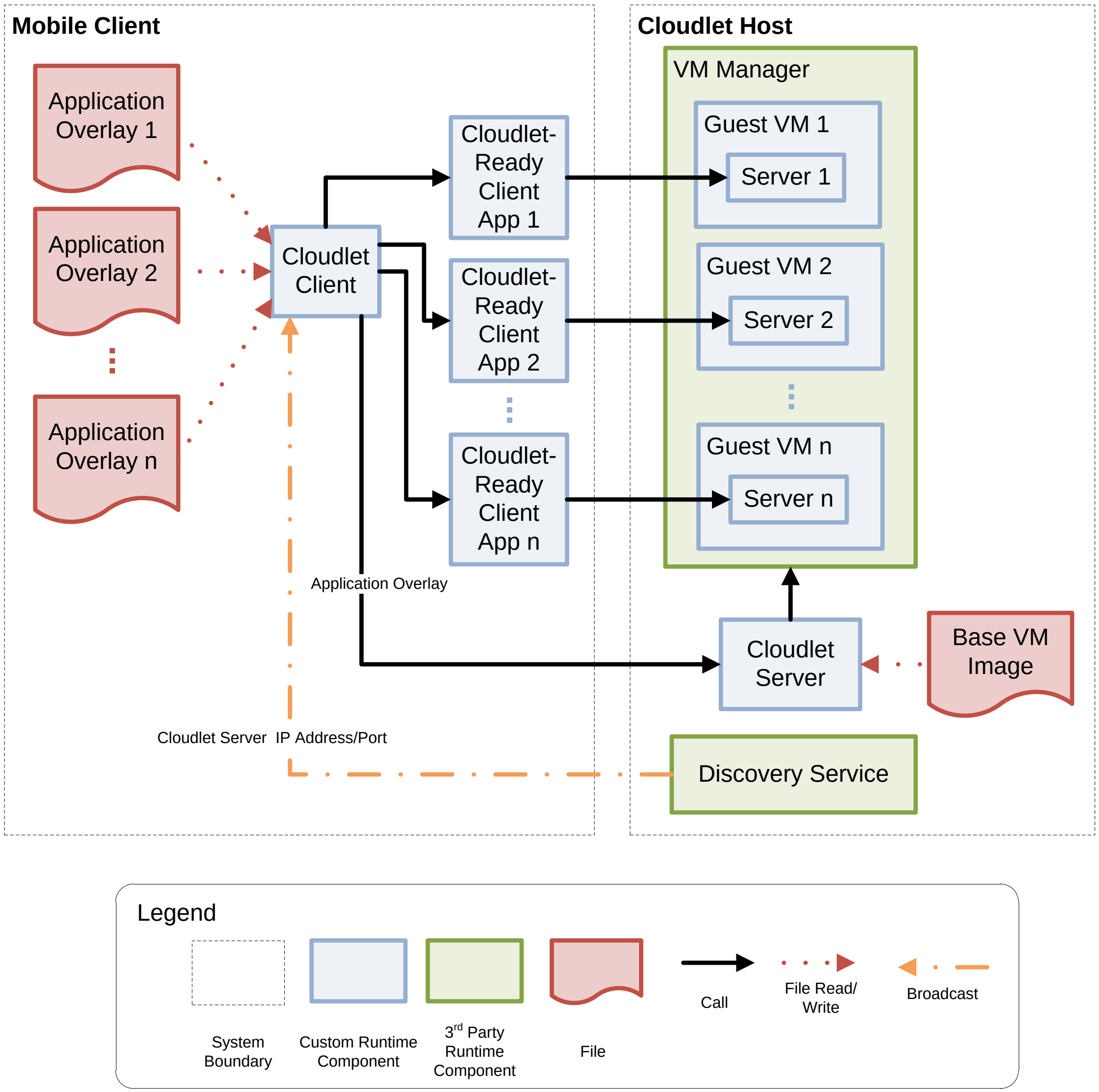
- Correspond to the server portion of a mobile app
- Created once, offline, by qualified personnel

Only constraint is that cloudlets need to store a copy of the same Base VM that was used for overlay creation

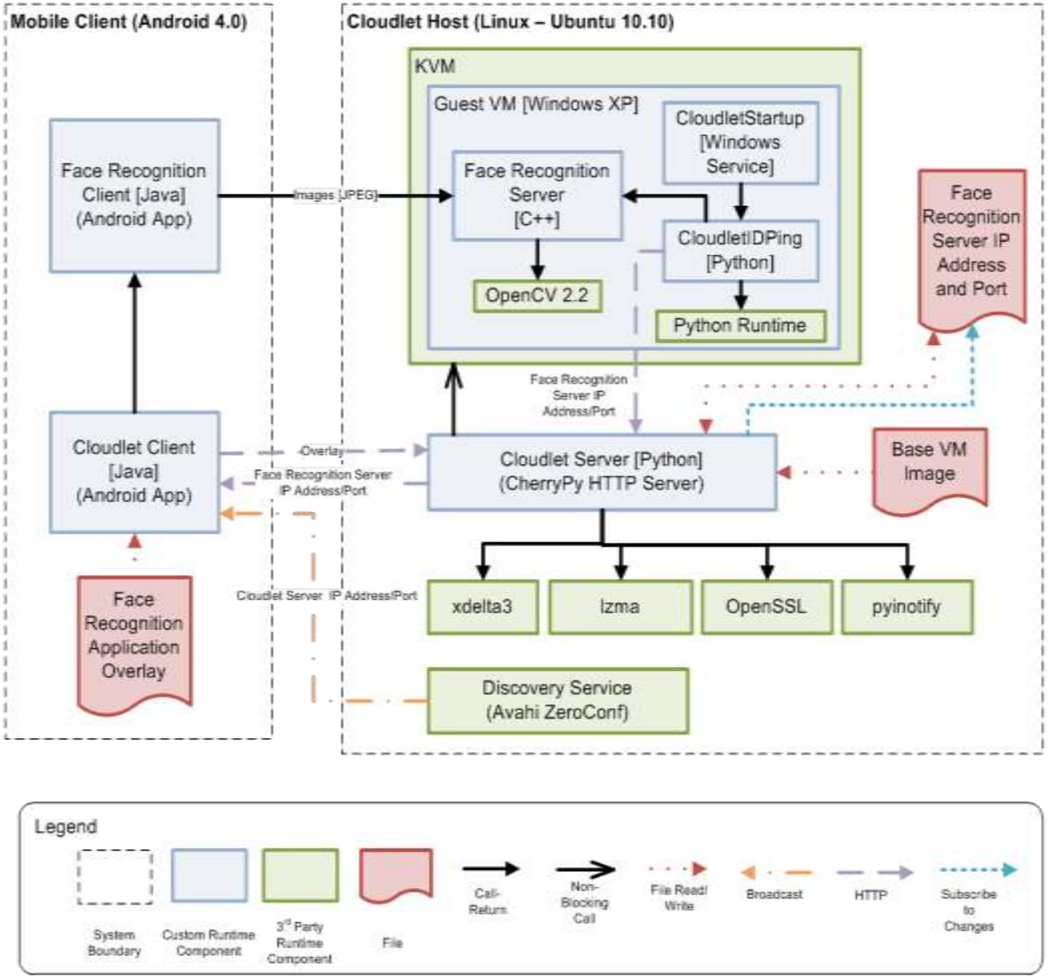
During execution

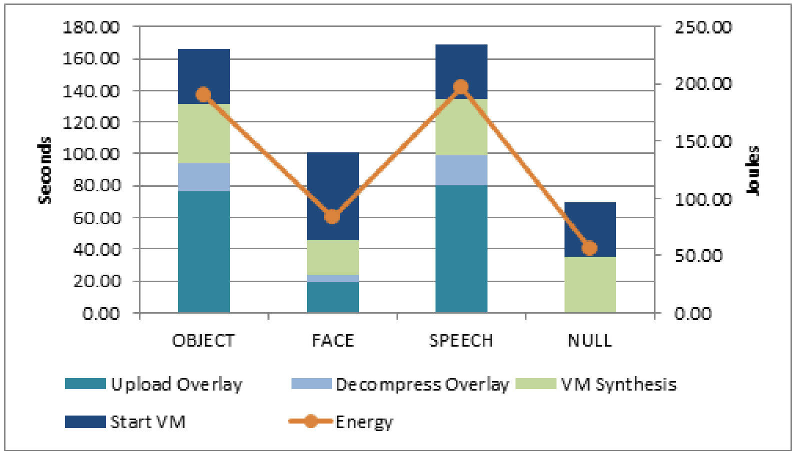
- Mobile device discovers available cloudlets
- Mobile device uploads application overlay to selected cloudlet
- Cloudlet applies application overlay to the Base VM and produces a Complete VM
- Cloudlet starts Complete VM which is now ready for application execution

Reference Architecture for Code Offload Based on VM Synthesis



Prototype 1: Initial Prototype





Application	Platform	Language	Application Size (MB)	Base VM Disk Image Size (MB)	VM Disk Image Overlay Size (MB)
OBJECT	Linux	C++	27.50	3546	165.32
FACE	Windows XP	C++	17.65	3073	43.55
SPEECH	Linux	Java	51.04	3546	176.23
NULL	Linux	N/A	N/A	3546	0.12

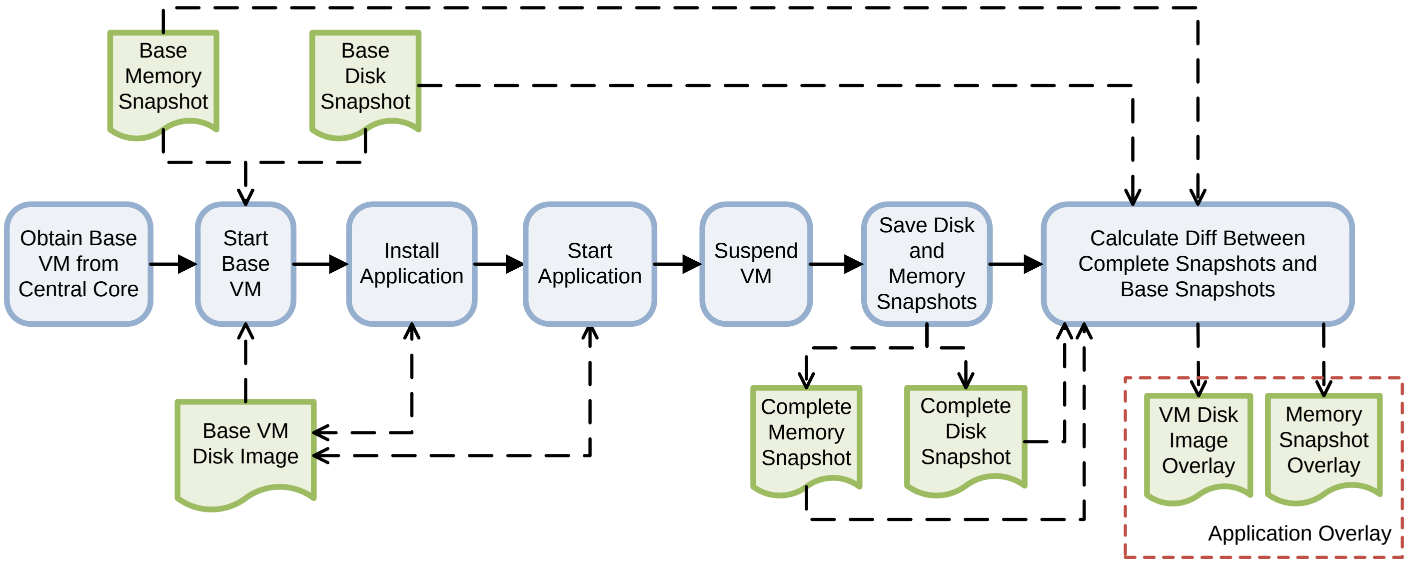
Prototype 1: Analysis

Limitations

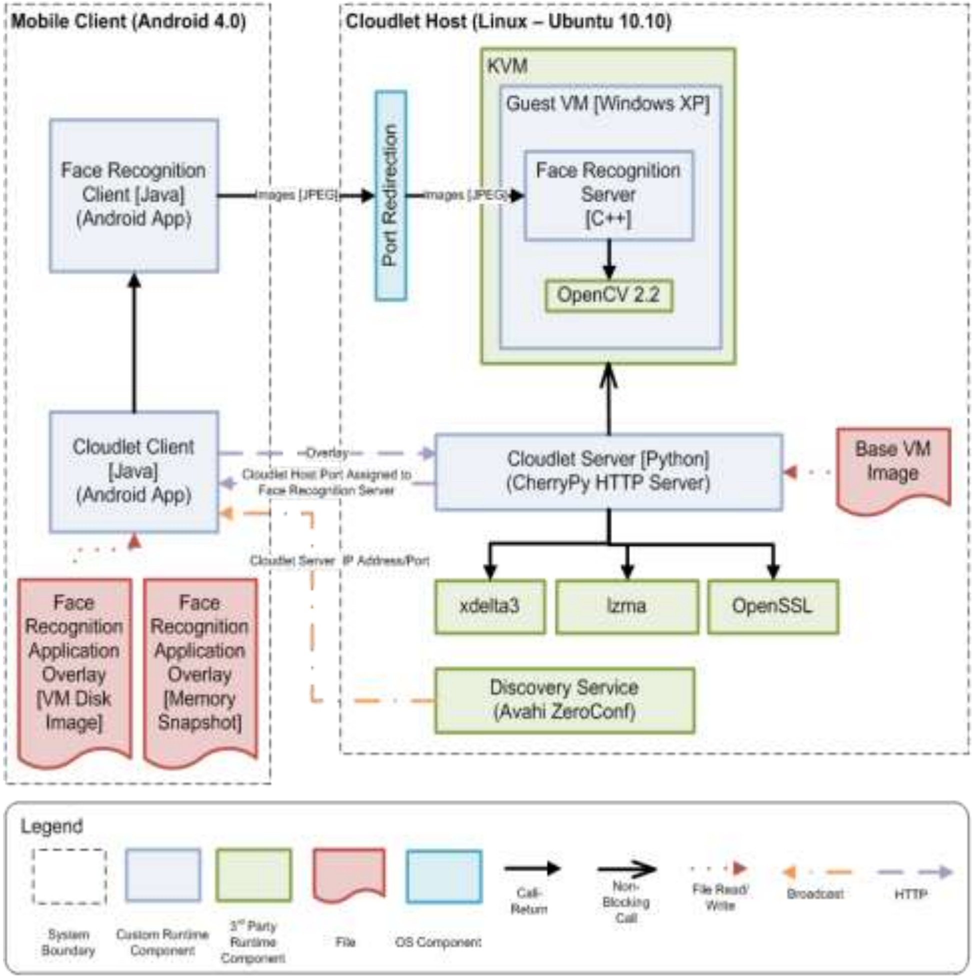
- Large overlays
- Long Start VM times
- Implementation complexity

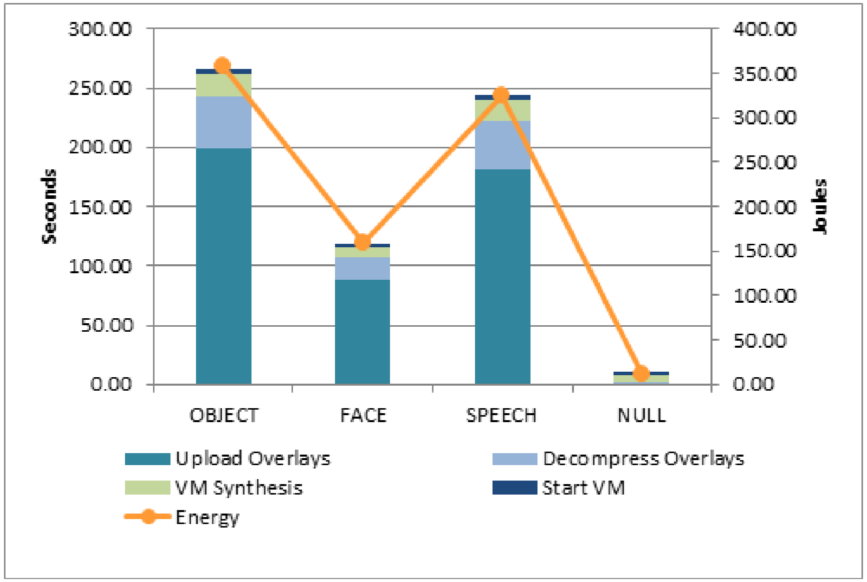
Major Changes for Prototype 2

- Disk image format: Changed from raw to qcow2
- Memory snapshot overlay plus disk-image overlay
- KVM in NAT mode and port redirection



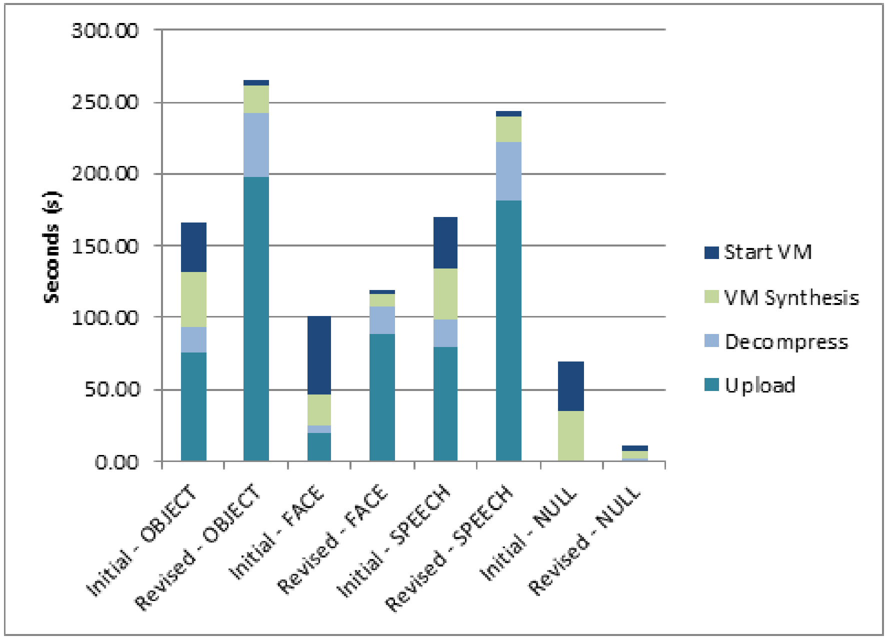
Prototype 2: Revised Prototype





Application	Base VM Disk Image qcow2 (MB)	Base Disk Snapshot qcow2 (MB)	Base Memory Snapshot (MB)	Compressed VM Disk Image Overlay (MB)	Compressed Memory Snapshot Overlay (MB)
OBJECT	3558	17	554	94	293
FACE	2421	15	278	71	101
SPEECH	3558	17	554	86	257
NULL	3558	17	554	1	3

Prototype 2: Analysis



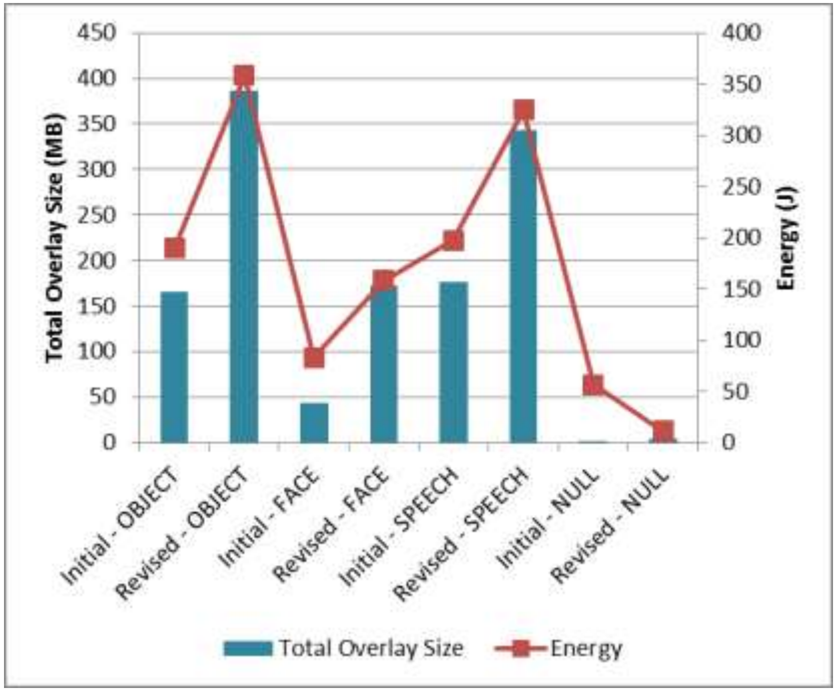
For the revised prototype to pay off, the efficiencies gained in VM Synthesis and Start VM would require supplementation with greater bandwidth

Advantages

- Shorter VM Synthesis and Start VM times
- Simple implementation

Limitation

- Very large overlays lead to increased battery consumption



Current and Future Work

Current work

- Application-level virtualization
  - Using static and dynamic analysis tools to create packages with all dependencies—early results show that packages are 20% the size of overlays
  - Tradeoff is anticipation of necessary "containers"
- Mobility-induced cloudlet handoffs to transfer state between cloudlets with minimal interruption to a moving user
  - Challenges have been on the networking side and not the actual VM migration

Future work

- Rapid VM synthesis
  - Extension of the discovery protocol to enable VM caching so that overlays do not always require transmission
  - Exploiting of multicore architecture to parallelize VM synthesis activities
- Cloudlet-selection mechanism that maps application needs to cloudlet characteristics exposed as cloudlet metadata during the cloudlet-discovery process

Summary

Cloudlets are discoverable, localized, stateless servers running one or more virtual machines (VMs) on which mobile devices can offload expensive computation

- Provide a general-purpose strategy for code offload and resource optimization in hostile environments
- Enhance processing capacity and conserve battery power while at the same time providing ease of deployment and application flexibility in the field

The two implementations of the proposed references architecture show the tradeoffs between overlay size, battery consumption and application-ready time

- Operationalization of the concept will require further reduction in overlay sizes and incorporation of strategies for minimizing or eliminating overlay transfer

Contact Information

Grace A. Lewis

Research, Technology and Systems Solutions (RTSS) Program
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

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.