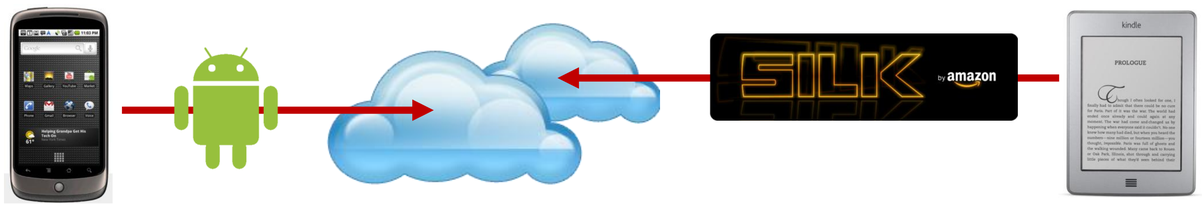
Enterprise Centric Offloading System

Aaron Gember & Aditya Akella *{agember,akella}@cs.wisc.edu*

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Application-Independent Offloading



Handheld users are running increasingly complex mobile applications with (1) high processing and energy demands, e.g., speech recognition, and (2) operations on sensitive data, e.g., health records. Application-independent offloading is a widely proposed solution to address handheld performance and energy demands without re-writing applications: MAUI [*Cuervo '10*], CloneCloud [*Chun '11*], Amazon Silk, etc. Offloading frameworks divide a mobile applications' execution between a handheld device and a compute resource.

We posit there are several key roadblocks limiting the adoption of existing offloading frameworks:

Privacy and trust: sensitive data in application state can be leaked in transit or on compute resources

Resource availability: applications may be offloaded to a diverse set of compute resources with varying availability and processing capabilities

Scalability: many handhelds with different goals (energy savings, latency improvements) must be able to simultaneously offload

Offloading for Enterprises

We focus on making offloading feasible for enterprises, which feature (1) increasing handheld usage for business, (2) a plethora of compute resources, e.g., idle desktops, dedicated servers, remote clouds, and (3) a single administrative domain.

How do we assign offloads to diverse compute resources to provide the most benefit for many handheld users?

Several factors must be considered in assigning resources:

- Varying user goals — latency improvement, energy savings
- Limited set of trusted compute resources
- Potentially changing resource availability
- Diverse processing capabilities
- Overhead of execution state transfer — depends on state size and latency/bandwidth to compute resource

ECOS Prototype

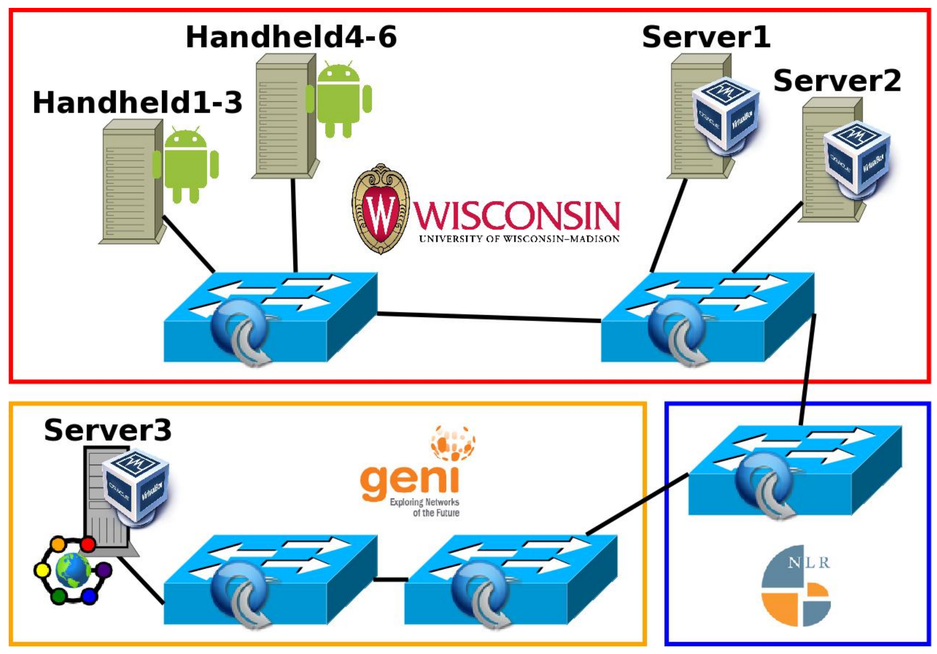
Enterprise Centric Offloading System (ECOS) is our in-progress prototype that addresses the roadblocks inhibiting the adoption of offloading. In practice, ECOS can be used in its entirety, or the security and resource assignment principles we employ can be used to augment any of the existing offloading proposals (CloneCloud, MAUI, etc.).

Our ECOS prototype currently has the following features

- Modified Android runtime environment capable of capturing, transferring, differencing, and restoring execution state
- Handheld offload agent responsible for "trapping" at offload points, requesting resources, and initiating offloads
- Resource offload agent responsible for reporting processing capabilities and managing offload runtime environments
- Logically central controller which allocates resources and populates switch flow tables based on policy-specified trust relationships, resource availability, and handheld users goals

Experimentation in GENI

We leverage GENI to (1) diversify our compute resource base and (2) identify key factors and mechanisms to include in our resource assignment algorithm. Android emulators running at Wisconsin serve as handheld devices, while Linux boxes at Wisconsin and ProtoGENI nodes at GPO serve as compute resources. All communication occurs over OpenFlow switches at Wisconsin, NLR, and GPO.



Our results have already identified the importance of

- Assigning offloads with different goals to different resources
- Preferring the same compute resource for subsequent offloads to reduce state transfer overhead

We are currently using experiments in GENI to understand

- How significantly throughput impacts offload overhead?
- Whether we can learn, based on execution times on compute resources, which offloads provided no benefits and adapt our offload and resource assignment decisions?