Demo: Mobile Application Resource Optimizer (ARO)

Feng Qian[1], Zhaoguang Wang[1], Alexandre Gerber[2],
Z. Morley Mao[1], Subhabrata Sen[2], Oliver Spatscheck[2]
[1]University of Michigan and [2]AT&T Labs Research

Categories and Subject Descriptors:
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – Wireless Communication;
C.4 [**Performance of Systems**]: Measurement Techniques

General Terms: Design, Measurement, Performance

Despite the popularity of mobile applications, their performance and energy bottlenecks remain hidden due to a lack of visibility into the resource-constrained mobile execution environment with potentially complex interaction with the application behavior. We design and implement ARO [1], mobile **A**pplication **R**esource **O**ptimizer, the first tool that efficiently and accurately exposes the cross-layer interaction to enable the discovery of inefficient resource usage.

System Overview. ARO consists of two components: the data collector and the analyzers. The data collector runs efficiently on a handset to capture information essential for understanding resource usage, user activity, and application performance. The collected traces are subsequently fed into the analyzers, which run on a PC, for offline analysis. We describe the workflow of ARO as outlined in Figure 1.

1. The ARO user invokes on her handset the data collector, which collects relevant data, *i.e.,* all packets in both directions and user input. ARO also identifies the packet-to-application correspondence. This is used to distinguish the *target application*, *i.e.,* the application to be profiled, from other applications simultaneously accessing the network.

2. The ARO user launches the target application and uses it as an end user. To obtain a representative understanding of the application studied, ARO can be used across multiple runs or by multiple users to obtain a comprehensive exploration of different usage scenarios of the target application.

3. The ARO user loads the ARO analysis component with the collected traces. ARO then configures the RRC analyzer with handset and carrier specific parameters, which influence the model used for analysis. The TCP, HTTP, and burst analyzers are generally applicable.

4. ARO then performs a series of analyses across several layers. In particular, the RRC (Radio Resource Control) state machine analysis accurately infers the radio states from packet traces so that ARO has a complete view of radio resource and radio energy utilization during the entire data collection period. ARO also performs transport protocol and application protocol analysis to associate each packet with its transport-layer functionality and its application-layer semantics. ARO next performs burst analysis, which utilizes aforementioned cross-layer analysis results, to understand the triggering factor of each short traffic burst, which is the key reason of low efficiency of resource utilization.

5. ARO profiles the application by computing for each burst (with its inferred triggering factor) its radio resource and radio energy consumption so that an ARO user can accurately identify and quantify the resource bottleneck for the application of interest.

Implementation and Demo Setup. We have fully implemented ARO for Android 2.2 (8.5K LoC). We apply ARO to more than 20 real Android applications each with at least 250K downloads from the Android market as of Mar 2011. ARO reveals that many of these very popular applications (*e.g.,* Fox News, Pandora, and EBay) have significant resource utilization inefficiencies that are previously unknown. In the demo, we will show step-by-step how to interactively use ARO to identify resource inefficiencies of smartphone applications by analyzing real traces (Figure 2). No special equipment is required other than a demo booth.

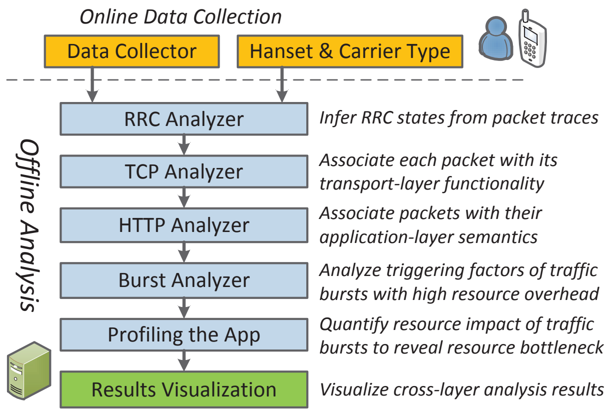
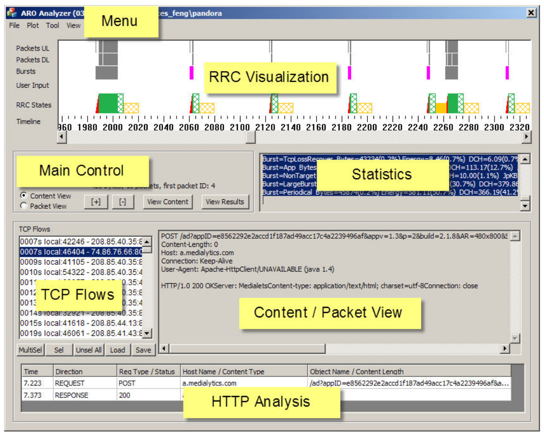

Online Data Collection
Data Collector
Hanset & Carrier Type
Offline Analysis
RRC Analyzer
Infer RRC states from packet traces
TCP Analyzer
Associate each packet with its transport-layer functionality
HTTP Analyzer
Associate packets with their application-layer semantics
Burst Analyzer
Analyze triggering factors of traffic bursts with high resource overhead
Profiling the App
Quantify resource impact of traffic bursts to reveal resource bottleneck
Results Visualization
Visualize cross-layer analysis results

Figure 1: The ARO System



```
Burst=TcpLossRecover Bytes=43234(0.2%) Energy=8.46(0.7%) DCH=6.09(0.7
Burst=App Bytes                    DCH=113.17(12.7%)
Burst=NonTarget                    H=10.00(1.1%) JpKB
Burst=LargeBurst                   (30.7%) DCH=379.86
Burst=Periodical Bytes=45874(0.2%) Energy=581.11(50.7%) DCH=366.19(41.2
```

```
0007s local:42246 - 208.85.40.35:8
0007s local:46404 - 74.86.76.66:80
0009s local:41105 - 208.85.40.35:8
0010s local:54322 - 208.85.40.35:4
0014s local:32921 - 208.85.40.35:8
0015s local:41618 - 208.85.44.13:8
0019s local:46061 - 208.85.41.43:8
```

```
POST /ad?appID=e8562292e2accd1f187ad49acc17c4a2239496af&appv=1.3&p=2&build=2.1.8&AR=480x800&
Content-Length: 0
Host: a.medialytics.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

HTTP/1.0 200 OKServer: MedialetsContent-type: application/text/html; charset=utf-8Connection: close
```

Time	Direction	Req Type / Status	Host Name / Content Type	Object Name / Content Length
7.223	REQUEST	POST	a.medialytics.com	/ad?appID=e8562292e2accd1f187ad49acc17c4a2239496af&a...
7.373	RESPONSE	200		

Figure 2: The ARO analyzer user interface

1. REFERENCES

[1] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Profiling Resource Usage for Mobile Applications: a Cross-layer Approach. In *Mobisys*, 2011.