Research Statement

Lenin Ravindranath Sivalingam

The mobile app ecosystem has grown at a tremendous pace, with billions of users, millions of apps, and hundreds of thousands of app developers. Mobile apps run across a wide range of network, hardware, location, and usage conditions that are hard for developers to emulate or even anticipate during lab testing. Hence, app failures and performance problems are common *in the wild*. Scarce resources and poor development support has made it more difficult for app developers to overcome these problems. My **research goal** is to **build systems** that make it radically easy for app developers to diagnose and improve their mobile apps.

To reduce user annoyance and survive the brutally competitive mobile app marketplace, developers need systems that (i) identify potential failures and performance problems before the app is released, (ii) diagnose problems after the app is deployed in the wild, and (iii) provide reliable app performance in the face of varying conditions in the wild. I have built systems that satisfy these needs. **VanarSena** [1] makes it easy to diagnose common problems in mobile apps before deployment, **AppInsight** [2] makes it easy to monitor mobile apps after deployment, and **Timecard** [3] allows mobile apps to adapt to conditions in the wild and provide consistent performance. To build these systems, I have actively combined ideas from different areas including networking, machine learning, software engineering, and programming languages.

These systems have already been successfully deployed and tested with thousands of mobile apps. The systems are built on top of a binary instrumentation framework that automatically rewrites app binary at bytecode level. Hence, using them requires **minimal effort** on part of the app developer. The systems include novel instrumentation techniques to automatically monitor and modify the runtime behavior of the app. To cope with the scarcity of resources, they include resource-aware mechanisms that incur **negligible overhead**. To make them **immediately deployable**, they are designed to require no modification to the OS or runtime. I briefly describe the key insights and challenges in each system below.

VanarSena [1] does automatic app testing by systematically emulating conditions from the wild to uncover failures and performance problems. With VanarSena, developers can easily identify and fix problems that can potentially occur in the wild, before the app is released. The techniques in VanarSena are driven by a study of 25 million real-world crash reports of Windows Phone apps reported in 2012. The analysis indicates that a modest number of root causes are responsible for many observed problems, but that they occur in a broad range of places in an app, requiring a wide coverage of possible execution paths. VanarSena adopts a gray box testing method, instrumenting the app binary to achieve high coverage. It uses dynamic monitoring techniques such as *hit testing* and *transaction tracking* to significantly improve the speed of testing. It runs several app "monkeys" in parallel to emulate user, network, and sensor data behavior, returning a detailed report of crashes and performance issues for common root causes.

VanarSena helps to significantly reduce common crashes and performance issues but is far from a complete solution to fix problems that occur in the wild. The mobile environment is complex and varying. To improve the quality of apps, there is also a need for developers to continuously monitor the behavior of apps after it is deployed. To achieve this, I built AppInsight.

AppInsight [2] does light-weight monitoring of mobile apps in the wild to pinpoint the root causes of failures and performance problems. AppInsight automatically instruments the app to monitor performance bottlenecks and failures in the app. Automatically monitoring the performance of mobile apps is challenging. To keep the UI responsive, mobile apps use a highly asynchronous programming model. Identifying performance bottlenecks and reasoning about failures in such code requires tracking causality across asynchronous boundaries. This challenging task is made even more difficult because of scarce resources available on the device such as CPU, battery, memory and network. In AppInsight, I developed a generic, low overhead instrumentation technique for monitoring the execution of asynchronous programs. Using this technique, AppInsight can track *user transactions* and accurately measure the user-perceived delays for interactions. AppInsight analyzes the collected data and provides detailed feedback to the developer about bottlenecks and failures in the wild. Using AppInsight, developers can iteratively improve the performance and reliability of their mobile apps.

AppInsight data from hundreds of apps in the wild shows that, the performance of a mobile app can be

highly variable. The variability comes from several factors such as network, GPS, hardware, etc. To improve user experience, it is important to provide fast and consistent performance in the face of such variability. To achieve this, I built Timecard.

Timecard [3] enables mobile apps to provide consistent performance by adapting to conditions in the wild. Timecard is specifically built for server-based mobile applications since user interactions that involve communication to a server have long and variable delays. To tightly control end-to-end delays in the face of variability, a transaction needs to adapt its processing time by trading-off either the quality of result or the amount of resources consumed. A natural place for such adaption is the server which typically does the core processing of the transaction. To adapt its processing time to manage the end-to-end delay, the server needs to know about the external delays: (i) the time that has elapsed since the user initiated the request in the app and (ii) the (remaining) time it will take for the app to receive an intended response over the network and then process it. Timecard accurately estimates and predicts these components and provides it to the server.

There are several challenges in automatically estimating the elapsed time and remaining time. Tracking elapsed time requires accurate and lightweight accounting across multiple, overlapping asynchronous threads that constitute the processing of a request on both the mobile app and the server. When the request reaches the server, the system must account for the clock difference between the client and the server. Predicting remaining time is difficult because it depends on many factors such as device type, network provider, response size, and TCP parameters. Timecard includes novel mechanisms to overcome these challenges. It consists of an automatic transaction tracking technique, a network-aware time synchronization mechanism, and data-driven prediction models. Using Timecard, servers can adapt their processing to provide consistent performance to mobile apps.

Impact: AppInsight has had wide impact. It has been successfully deployed at Microsoft and has helped many developers improve their Windows Phone apps. The instrumentation and tracing framework in AppInsight is being used by at least a dozen projects inside Microsoft. For instance, the AppCompat team in the Windows Phone group uses a version of AppInsight on thousands of apps on a daily basis. Also, there have been efforts in academia and industry to port AppInsight to other platforms. VanarSena is being run as an experimental service and has been used to test thousands of existing apps in the app store. It uncovered problems with many apps including those that were not previously reported. VanarSena has generated significant interest inside Microsoft to test Windows Phone apps and is currently being transferred to product groups. Timecard and its components are being used by multiple research projects to enable servers to accurately measure and predict external delays.

Other Mobile Systems

My interest in building systems to *improve performance* extends beyond applications. The uncontrolled mobile environment not only affects application performance, but also affects the performance of network protocols. To improve network performance, I built a **context-aware network architecture** [4, 14] that integrates information from external sensors into the networking stack thereby enabling protocols to adapt to the context of the device.

Also, my interest in building systems to *improve applications* extends beyond performance and reliability. I have designed and built systems that have addressed many other challenges app developers face today. **Procrastinator** [5, 13] enables mobile apps to trade-off between performance and network cost. **Code In The Air** [7, 15, 16] provides a programming framework to easily build distributed sensing applications. **SmartAds** [6] brings contextual ads to mobile apps, increasing the relevancy of ads. **Social+** [11] provides an in-app socialization platform to increase the engagement of users inside a mobile app. **VTrack** [8] and **CTrack** [9] improves the accuracy and energy efficiency of continuously location tracking.

Research Directions

For the legion of amateur app developers with fewer resources at hand, I want to continue building systems that significantly reduces the barrier for creating good mobile applications. These systems include (i) server infrastructure support for developers to easily build and maintain cloud-based apps, (ii) a framework for in-the-wild A/B testing that enables developers to optimize their app after it is deployed, (iii) a system that enables developers to do targeted updates to apps [12], and (iv) better programming abstractions and tools that allows developers to not only reason about performance but also reason about other optimization goals such as reducing energy consumption and reducing network cost. There are several research challenges in building these systems and I am excited about solving them.

Wearables: The next big wave of consumer mobile devices will be wearables such as glasses, watches and clip-ons. These devices bring exciting new possibilities as well as new challenges for app developers. As wearables

gain popularity, we are also going see the arrival of continuous monitoring high datarate applications. Examples include augmented reality applications that can overlay information about the objects in the view of the glass and Siri-like mobile assistants that can continuously interact with users. To enable these real-time applications, large amount of sensor data needs to be continuously shipped to the cloud where it needs to be processed within few hundred milliseconds. With resource-constrained wearables, we are in fact going have a three-tier model where the wearable communicates to the smartphone which in turn communicates to the cloud. With varying environment and network conditions, building the processing pipeline for such high responsive apps across the tiers is going to be a challenging task. I want to build programming and infrastructure support to enable such applications to continuously adapt the processing pipeline and explore the inherent trade-offs between responsiveness, accuracy and resources consumed.

Large-scale analytics: Large-scale analytics data collected from thousands of apps and the wide range of sensors will be the key for developers to optimize and personalize mobile user experiences. Given the scarce resources on mobile devices, data needs to be efficiently collected across all users and the collection pipeline needs to be continuously adapted based on the resources available. Borrowing ideas from compressive sensing, I want to build a generic data collection framework that can be used to collect the *optimal* amount of data across users. Further, I want to build systems to efficiently process this data and infer metrics that enables developers to improve and personalize the user experiences in a fine-grained and context-aware manner. I am also interested in the problem of big-data visualization where the goal is to efficiently present such large-scale data to developers in a way that they can quickly derive value out of it.

App-aware systems: While it is important for mobile applications to be context aware, I also believe that, it is important for mobile systems to be application aware. Mobile apps differ widely in their runtime behavior and system components need to adapt to application behavior to perform efficiently. For instance, our recent analysis on thousands of mobile apps shows that mobile apps have very different network access characteristics. And there is no single network protocol setting that can perform well for all apps. There is need for protocols to adapt based on the behavior of the application to maximize performance. I believe that, it applies to other system components such as caching, garbage collection, radio wakeup etc. To this end, I want to explore the idea of app-aware systems that can automatically analyze the runtime behavior of an app and adapt the system parameters based application behavior. I want to build a framework (similar to the VanarSena) that can dynamically analyze the behavior of the app based on conditions in the wild and find the optimal system settings for the app.

Health-aware systems: I am also excited to build systems at the intersection of mobile computing and other fields. I believe that mobile devices such as wearables, being in constant physical touch with the user, have the ability to pave way for new health care applications by integrating health sensors such as heart rate, blood pressure and skin conductivity monitors. I am particularly interested in tapping these sensors to build health-aware systems where the mobile system adapts to the current physiological state of the user. I want to enable scenarios that include adapting the UI state based on user fatigue, adapting the speed of a game based on the users heart rate, and fingerprinting changes in physiological indices to predict and prefetch application state. I want to pursue ideas from different fields such as psychology and HCI to build such innovative mobile systems.

References

[1] L. Ravindranath, S. Nath, J. Padhye, and H. Balakrishnan, "Automatic and Scalable Fault Detection for Mobile Applications," in *MobiSys*, 2014.

[2] L. Ravindranath, J. Padhye, S. Agarwal, R. Mahajan, I. Obermiller, and S. Shayandeh, "AppInsight: Mobile App Performance Monitoring in the Wild," in *OSDI*, 2012.

[3] L. Ravindranath, J. Padhye, R. Mahajan, and H. Balakrishnan, "Timecard: Controlling User-Perceived Delays in Server-Based Mobile Applications," in *SOSP*, 2013.

[4] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden, "Improving Wireless Network Performance Using Sensor Hints," in *NSDI*, 2011.

[5] L. Ravindranath, S. Agarwal, J. Padhye, and C. Reiderer, "Procrastinator: pacing mobile apps usage of the network," in *MobiSys*, 2014.

[6] S. Nath, F. Lin, L. Ravindranath, and J. Padhye, "SmartAds: Bringing Contextual Ads to Mobile Apps," in *MobiSys*, 2013.

[7] L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code In The Air: Simplifying Sensing and Coordination Tasks on Smartphones," in *HotMobile*, 2012.

[8] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan, "VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones," in *SenSys*, 2009.

[9] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, "Accurate, Low-Energy Trajectory Mapping For Mobile Devices," in *NSDI*, 2011.

[10] P. K. Gunda, L. Ravindranath, C. A. Thekkath, Y. Yu, and L. Zhuang, "Nectar: Automatic Management of Data and Computation in Data Centers," in *OSDI*, 2010.

[11] L. Ravindranath and P. Bahl, "A Platform for In-App Socialization."

[12] A. Cheung, L. Ravindranath, E. Wu, S. Madden, and H. Balakrishnan, "Mobile Applications Need Targeted Micro-Updates," in *APSys*, 2013.

[13] L. Ravindranath, S. Agarwal, J. Padhye, and C. Reiderer, "Give in to procrastination and stop prefetching," in *HotNets*, 2013.

[14] L. Ravindranath, C. Newport, H. Balakrishnan, and S. Madden, ""Extra-Sensory Perception" for Wireless Networks," in *HotNets*, 2010.

[15] L. Ravindranath, T. Chen, A. Sivaraman, H. Balakrishnan, and S. Madden, "Code in the Air: Simplifying Tasking on Smartphones," in *MobiSys (demo)*, 2012.

[16] T. Kaler, J. P. Lynch, T. Peng, L. Ravindranath, A. Thiagarajan, H. Balakrishnan, and S. Madden, "Code in the Air: Simplifying Sensing on Smartphones," in *SenSys (demo)*, 2010.