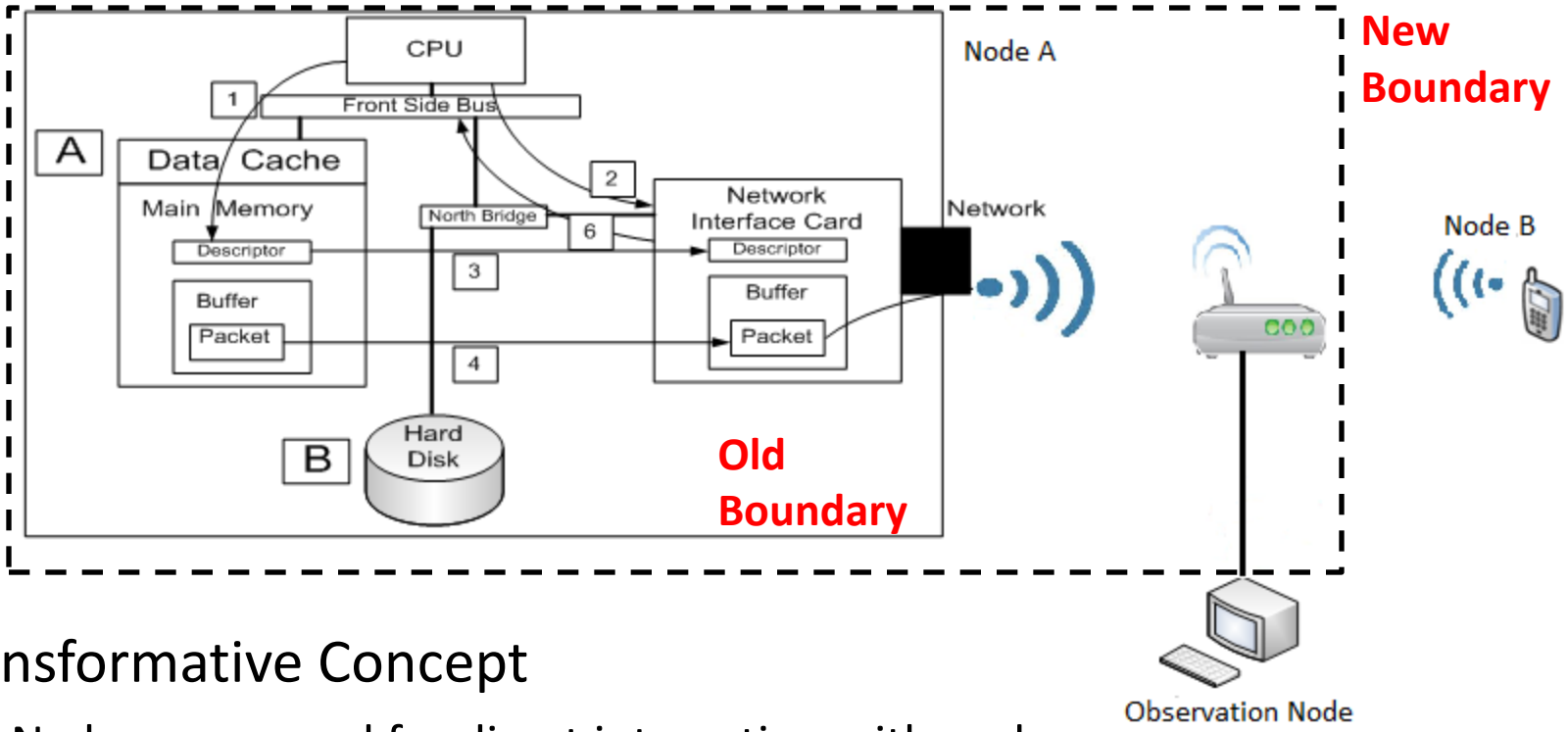
Using Network Traffic to Remotely Identify the Type of Applications Executing on Mobile Devices

Lanier Watkins, PhD
LanierWatkins@gmail.com

Outline

- Introduction
- Contributions and Assumptions
- Related Work
  - CPU and Memory Utilization Inference Technique
  - Mobile Device Management Software
- Feasibility Study
- Ground-Truth Analysis
- Remotely Identifying the Type of Applications Executing on Mobile Devices
- Results
- Summary
- Future Work

Introduction: Extending the Node's Boundary Into the Network

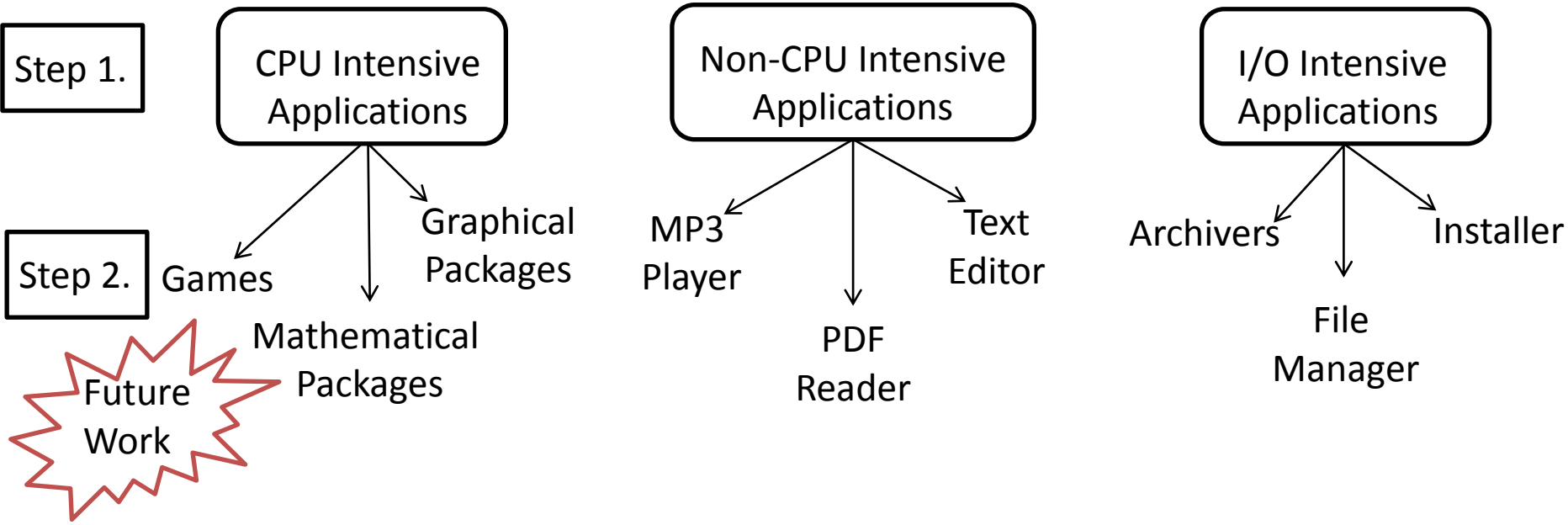


- Transformative Concept
  - No longer a need for direct interaction with nodes
  - Demonstrated by remotely identifying types of applications running on mobile devices
- Information Leakage (IL)
  - Nodes communicating on a network unknowingly give off useful information
  - We use IL to determine hardware state via extracted delay information

Introduction: Applicable Mobile Device Security Applications

Our Work Is Initial Step Toward

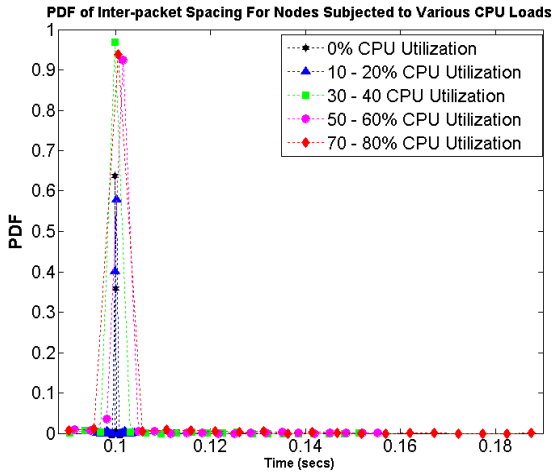
- Network-Based Application White/Black Listing
- Network-Based Anomaly Detection (Usage)
  - No daemons on mobile devices
  - Ability to provide visibility into device when non-network traffic generating applications are executing
  - Identifies high-level group to which applications belong (First Step)



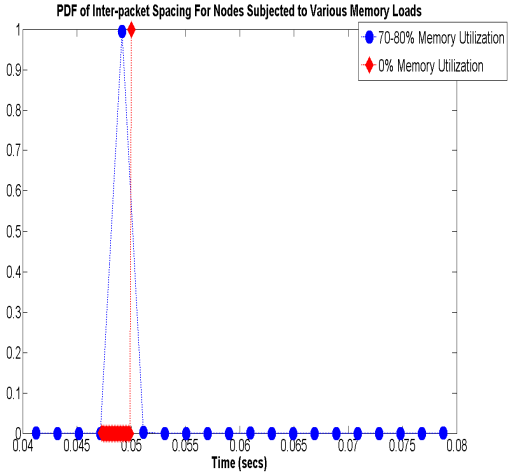
Contributions and Assumptions

- Contributions
  - Network-based method
    - Requires no daemon on device
      - Low over-head
      - Less software on a node makes it more secure (lowers attack surface)
  - Evaluates top mobile device vendors and Identifies Android-based devices as target
  - Demonstrates ability to remotely identify types of applications running on mobile devices over Wifi
- Assumptions
  - One application executing at a time
    - Due to memory constraints mostly services run in background which are triggered by certain events (**Future Work**)
  - Excellent Wifi strength
    - Degraded signal strength lowers correct classification
  - No user interaction
    - Result is spikes in CPU speed, which we believe this can be filtered out (**Future Work**)
  - Three broad categories of applications
    - Since governor sampling rate is on order of microseconds, there may be enough information available to identify specific applications (**Future Work**)

Related Work: Inferring CPU and Memory Utilization For Resource Discovery



CPU Utilization: PDF for 100 ms CBR UDP Network Traffic from available and unavailable nodes



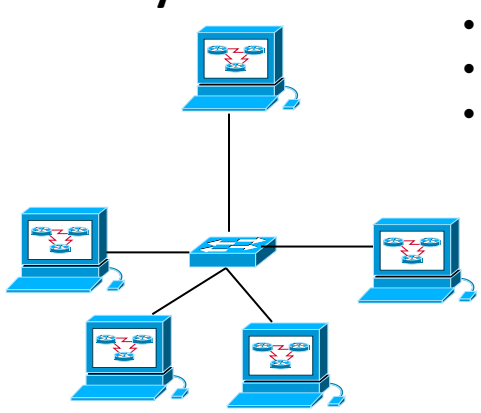
Memory Utilization: PDF for 50 ms CBR UDP Network Traffic from available and unavailable nodes

- Increased utilization (e.g., CPU or Memory) affects a node's ability to send network traffic at a constant rate
- A method can be devised to remotely identify heavily used nodes
  - Opportunity to reduce message overhead
  - Opportunity to reduce daemons executing on compute nodes

Related Work: Inferring CPU and Memory Utilization For Resource Discovery

- Deterlab used to emulate 50 node cluster
- Context switching was driver of detectable delays due to CPU utilization
- Paging was driver of detectable delays due to memory utilization

MPI-based Asymmetric cluster

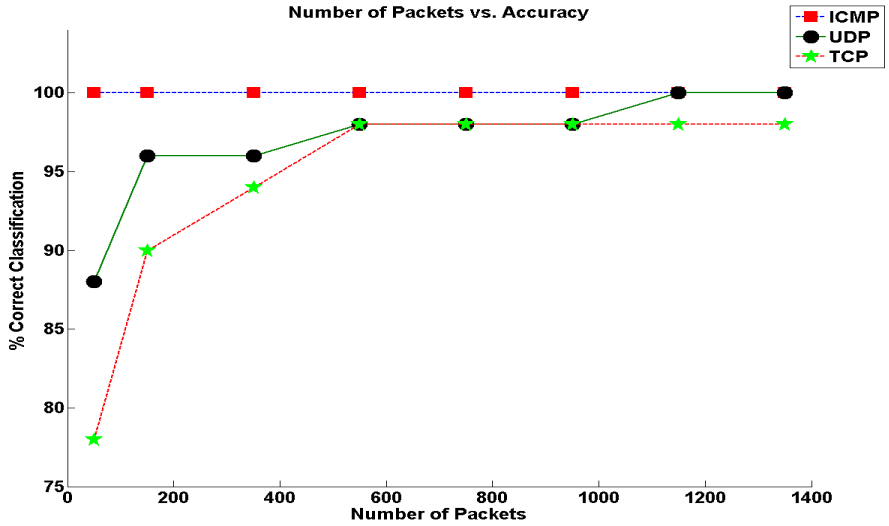


- Queue Manager
- Scheduler
- Resource Manager

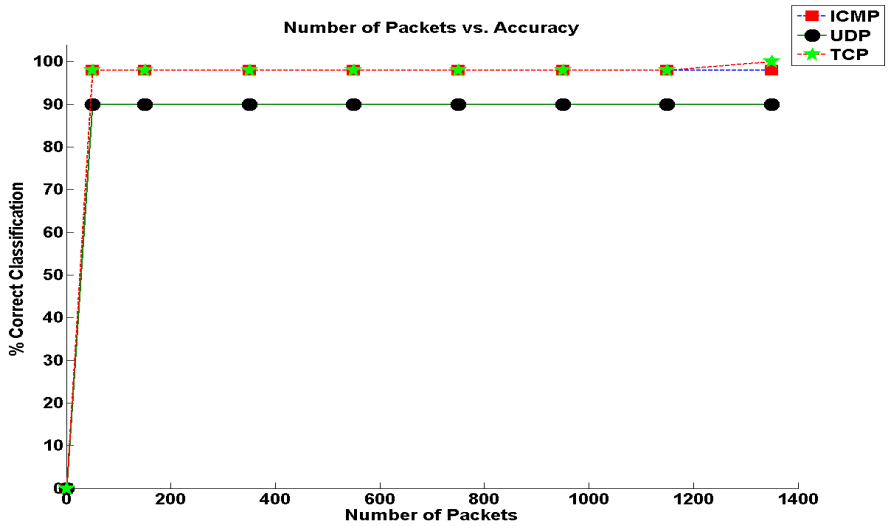
Head Node

Compute Nodes

CPU Inference [1]



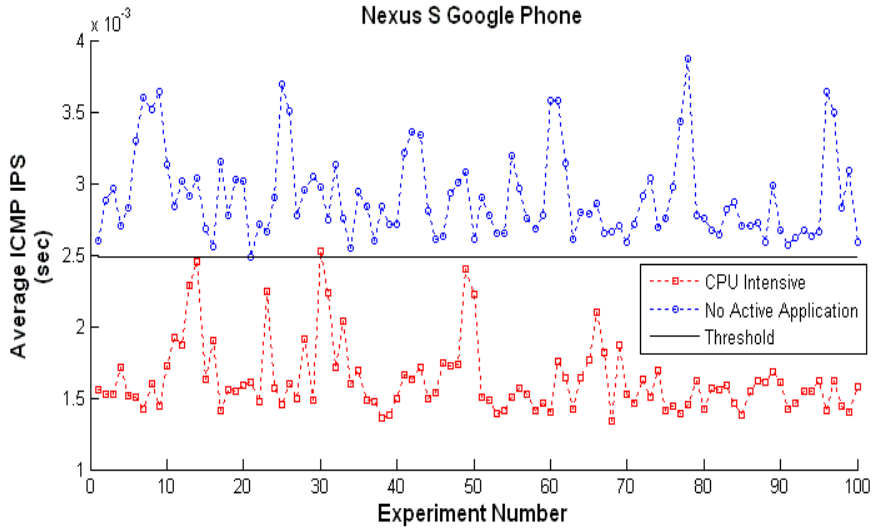
Memory Inference [2]



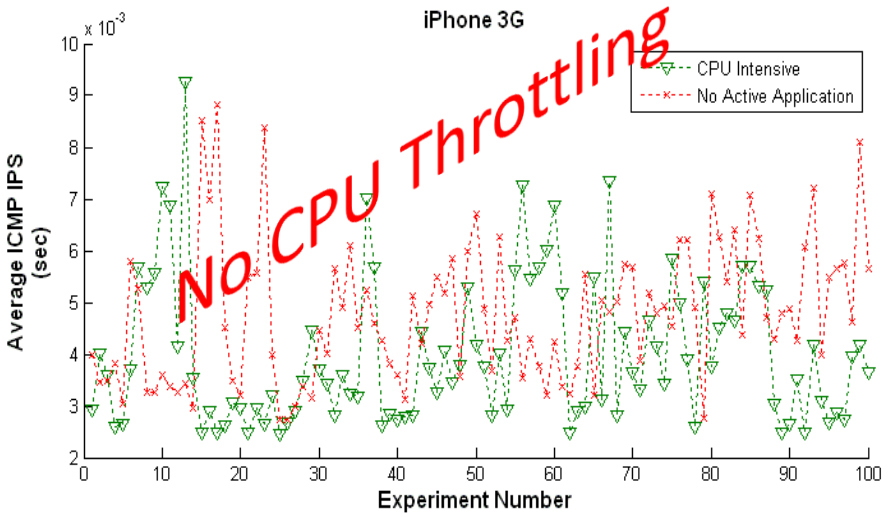
Related Work: Mobile Device Management Software

- Examples:
  - Airwatch [3], Good [4] , and MobileIron [5]
- Pros:
  - Can provide low-level application and process information
- Cons:
  - Host-based software
    - Could be compromised
      - Anti-virus host-based software can been circumvented [6]
    - High over-head
      - Daemon installed on every mobile device

Feasibility Analysis: Can CPU Speed Be Inferred? If So, By Which Wireless Devices?



Android OS 4.2/Google Phone:
200 experiments, successful attempts at discerning between 100 experiments with a high CPU application and 100 experiments with an idle CPU.

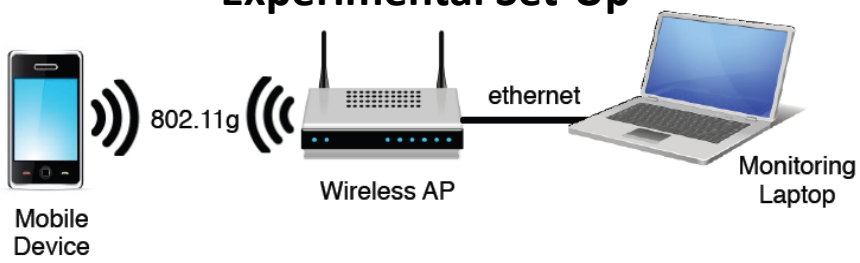


Apple iOS 4.2/iPhone 3G:
200 experiments, failed attempts at discerning between 100 experiments with a high CPU application and 100 experiments with an idle CPU.

Experimental Procedure

- Ping device 100 times with idle CPU and 100 times with busy CPU
- Capture ICMP reply timestamps
- Calculate average inter-packet spacing
- Repeat each experiment 100 times
- Choose discerning threshold

Experimental Set-Up



Terminology: What is CPU Throttling

Variables	Description
governor	Stores the name of the current governor (determines CPU needs of device, e.g., **ondemand**, performance, userspace)
up_threshold	Stores the maximum CPU load percentage allowable before the governor scales the CPU load up to the next level
sampling_rate	Stores the rate at which the governor samples the present CPU load
CPU_{min}	Stores minimum CPU speed
CPU_{max}	Stores maximum CPU speed

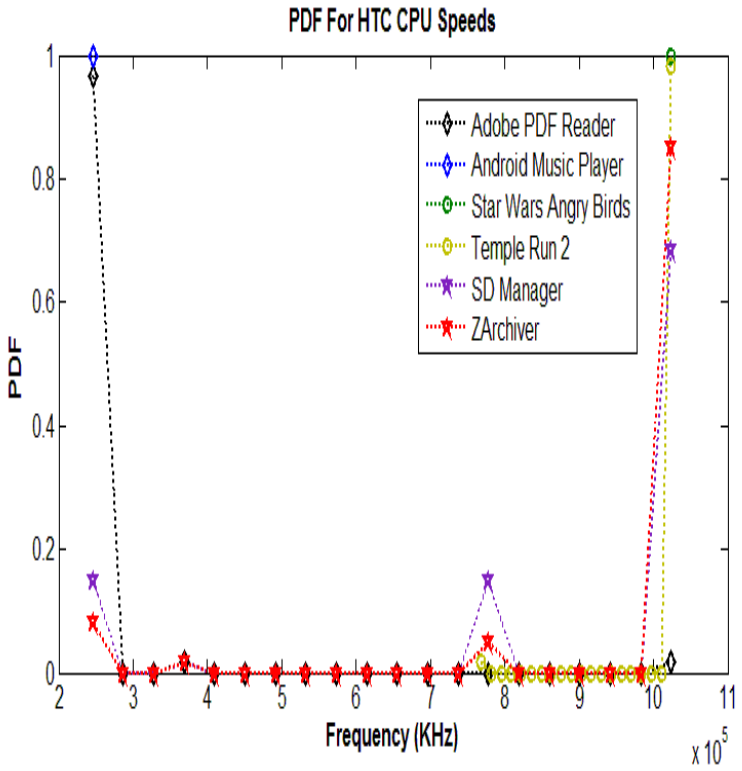
- Default governor for Android devices is ondemand [7]
- Ondemand governor throttles (moves up and down) due to the needs of foreground processes

Ground-Truth Analysis: Do Applications Actually Influence Mobile Device CPU Speed?

CPU Speed Comparison for 6 Applications Using HTC

User Applications		CPU Speed		
		Min	Max	Average
Non-CPU Intensive	Adobe Reader	245MHz	1GHz	260MHz
	Music Player	245MHz	245MHz	245MHz
CPU Intensive	Angry Birds	1GHz	1GHz	1GHz
	Temple Run	1GHz	1GHz	1GHz
I/O Intensive	ZArchiver	245MHz	1GHz	935MHz
	AppMgr III	245MHz	1GHz	858MHz

***None of these applications produce network traffic**



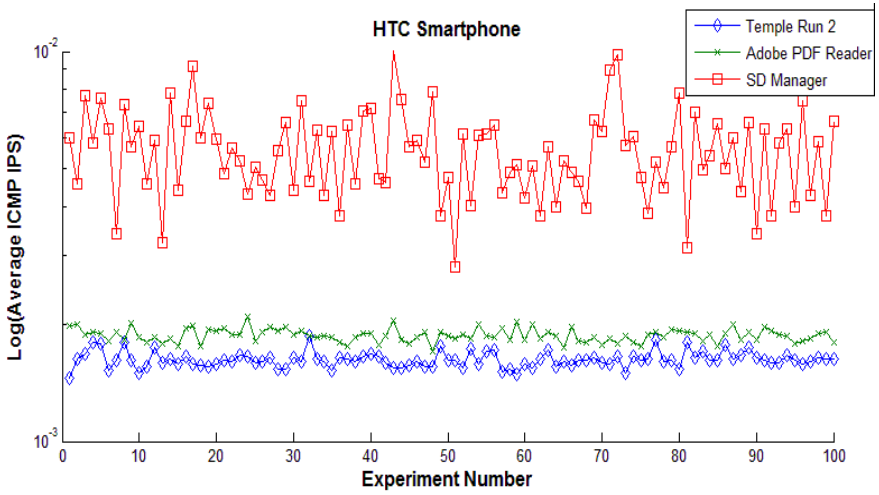
Observations

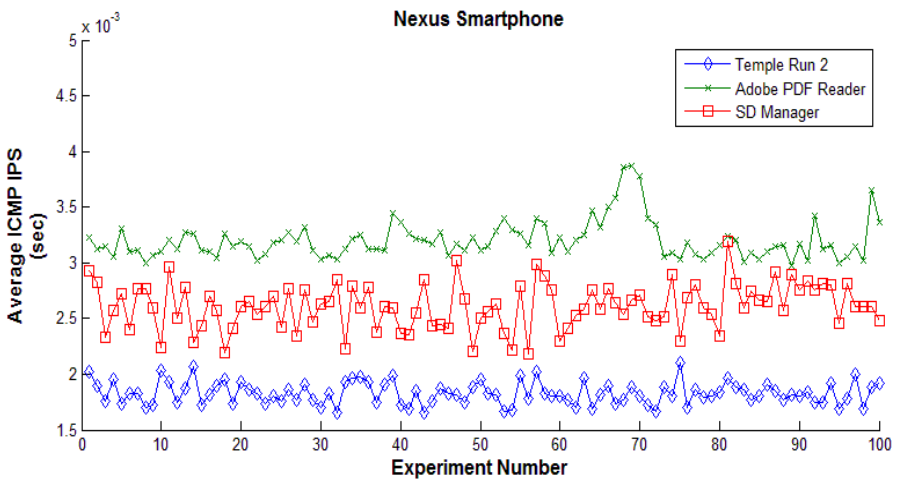
- CPU intensive and non-CPU intensive apps CPU speeds are as expected

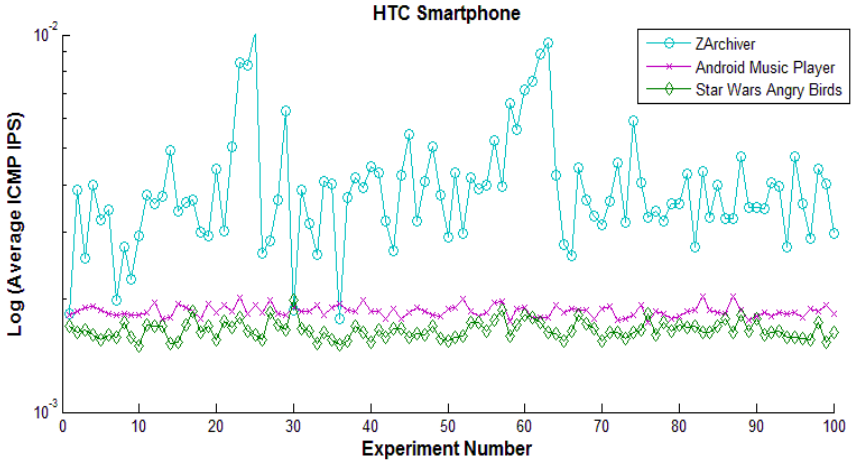
- I/O intensive apps induce a significant CPU load and consume I/O resources

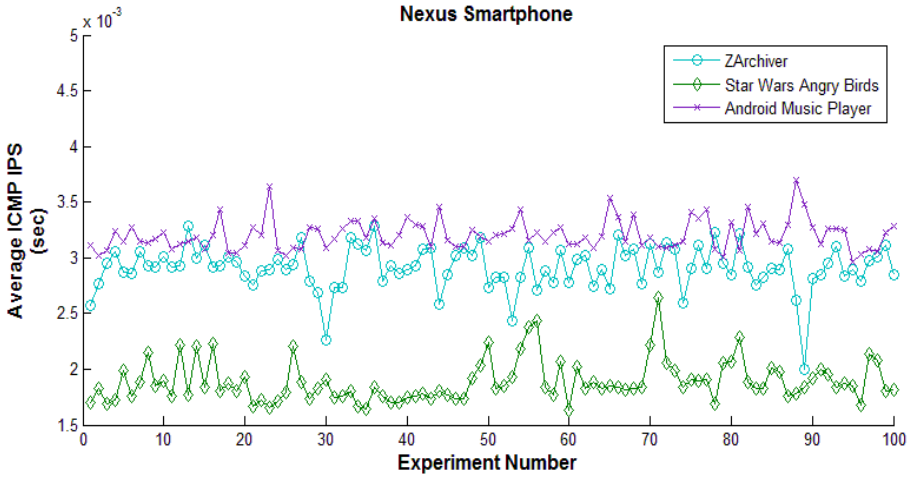
Experimental Set-Up

Extending Node Boundaries Into Network: Remotely Identifying User Application Type



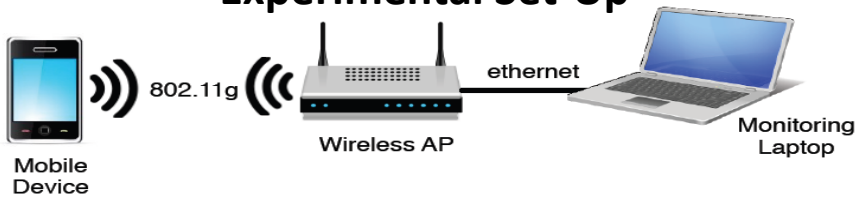






- CPU intensive have lowest IPS for both
- I/O intensive apps on HTC yield largest IPS
- HTC has removeable SD Card, Nexus internal SD Card [8]
- I/O intensive apps on Nexus yield 2nd largest IPS

Experimental Set-Up



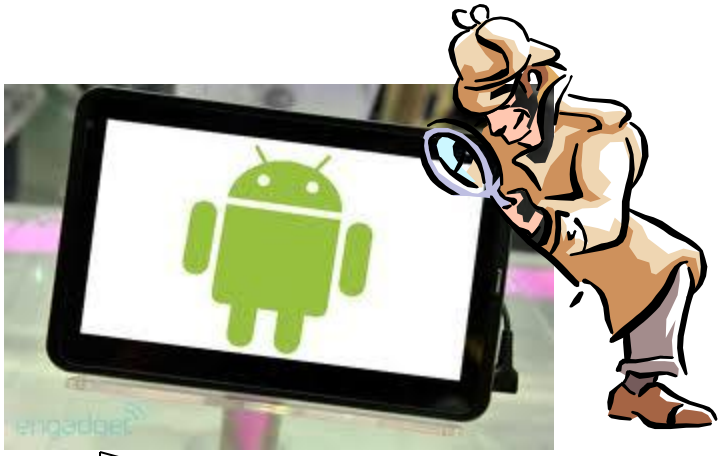
Extending Node Boundaries Into Network: Remotely Identifying User Application Type

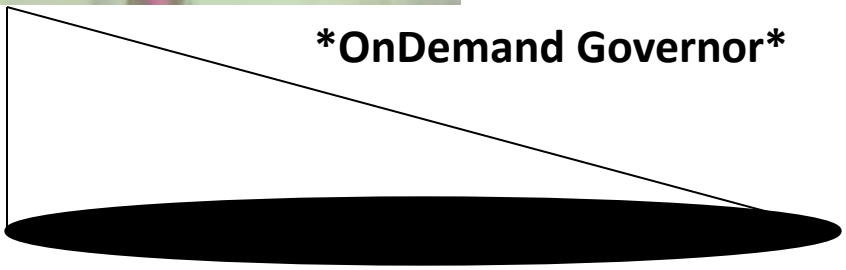
Average of Average IPS For 6 Applications

User Applications		HTC Average IPS	Nexus Average IPS
Non-CPU Intensive	Adobe Reader	1.9 ms	3.2 ms
	Music Player	1.9 ms	3.2 ms
CPU Intensive	Angry Birds	1.7 ms	1.9 ms
	Temple Run	1.6 ms	1.8 ms
I/O Intensive	Zarchiver	4.0 ms	2.9 ms
	AppMgr III	5.6 ms	2.6 ms

- Average of 100 experiments (Avg. IPS) per app per device suggests
  - These 6 apps can be grouped into types per device by Avg. IPS
  - Primary reason is that Avg. IPS is affected by:
    - CPU intensive apps throttling CPU high
    - non-CPU intensive apps throttling CPU low
    - I/O intensive apps throttling up CPU and claiming I/O resources
- It should be possible to use machine learning to perform this grouping

Theory Supported By Empirical Evidence: In Android-Based Wireless Devices, CPU Speed Affects Network Traffic





Time	t_1	t_2	t_3	…	t_n
Interrupt	I_1		I_2	…	I_n
Scheduler	P_1	P_2	P_3	…	P_n
CPU Speed	S_1	S_2	S_3	…	S_n

Peeking Inside Android Devices

- ICMP Request interrupt causes currently executing process to be preempted
- We suspect that network traffic sending kernel subroutines are executed faster or slower depending on the CPU speed of device at time of preemption
- Empirical evidence support theory that, busy mobile devices respond faster (i.e., network traffic), and less busy mobile devices respond slower
- Empirical evidence supports theory that I/O intensive processes affect CPU speed and we suspect that these processes claim external I/O resources which can greatly (HTC) or somewhat (Nexus) delay network traffic

Extending Node Boundaries Into Network: Analysis of Empirical Data And Results

Neural-Fuzzy Classifier (NFC) Confusion Matrix for 3 HTC App Types

95%

Actual Class \ Predicted Class	Non-CPU	CPU	IO
Non-CPU	97	3	0
CPU	11	89	0
IO	2	0	98

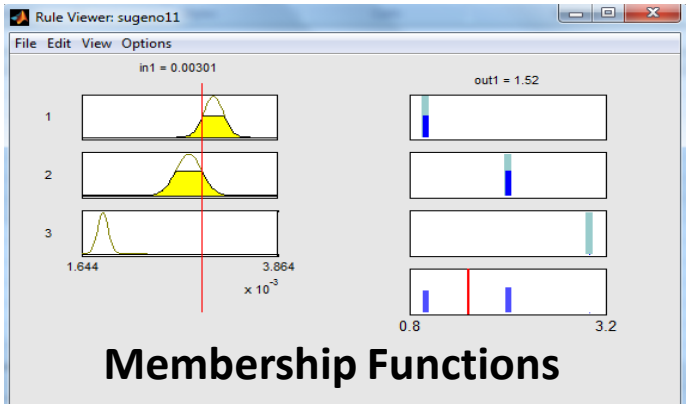
Neural-Fuzzy Classifier (NFC) Confusion Matrix for 3 Nexus App Types

93%

Actual Class \ Predicted Class	Non-CPU	CPU	IO
Non-CPU	98	0	2
CPU	0	92	8
IO	12	0	88

Fuzzy Rules

if input in MF1 then output is class 1
if input in MF2 then output is class 2
if input in MF3 then output is class 3

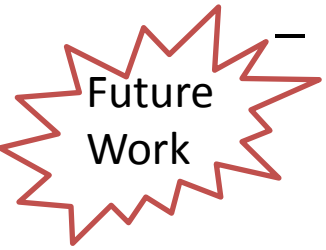


Membership Functions

- NFC [9] trained with the first 50 experiments per application per device (300 patterns)
- Fuzzy rules and membership functions created from training
- NFC tested on remaining 50 experiments per application per device (300 patterns)
- Experiments classified based on firing strength of fuzzy rules

Extending Node Boundaries Into Network: Further Discussion On Dataset

- Note, NFC is only trained and tested on 1 feature of the dataset
- We could probably increase the percent of correct classification or even uniquely identify each of the six applications by using more features in the dataset
  - Since the *sampling_rate* is on the order of microseconds, it is possible there is information hidden in the dataset which could be revealed by using signal processing (e.g., wavelet transform, fourier transform) techniques to pre-process the data.



Extending Node Boundaries Into Network : Summary

- Android-based mobile devices send network traffic faster or slower depending on the foreground application running on the device
  - The driver is CPU throttling (On-demand Governor)
- This empirical truth is exploitable such that the types of applications (i.e., non-CPU intensive, CPU intensive and I/O intensive) executing in the foreground can be remotely identified
  - Machine-learning techniques can be trained to accomplish this goal

Extending Node Boundaries Into Network : Future Work

We will investigate the below extensions to our work:

- Multiple applications executing simultaneously on device
  - Due to memory constraints mostly services run in background which are triggered by certain events
- User interaction with mobile device
  - Result is spikes in CPU speed, which we believe this can be filtered out
- Identifying specific applications
  - Since governor sampling rate is on order of microseconds, there may be enough information available to identify specific applications

Questions?

References

1. Lanier Watkins, William H. Robinson, Raheem Beyah, “A Passive Solution to the CPU Resource Discovery Problem in Cluster Grid Networks.” In IEEE Transactions on Parallel and Distributed Systems (TPDS), December 2011.
2. Lanier Watkins, William H. Robinson, Raheem Beyah, “A Passive Solution to the Memory Resource Discovery Problem in Grid Computing.” In IEEE Transactions on Network and Service Management (TNSM), December 2010.
3. Airwatch Website Accessed February 2013:
   http://www.air-watch.com/solutions/mobile-device-management
4. Good Website Accessed February 2013:
   http://www1.good.com/products/mobile-manager
5. Mobile Iron Website Accessed February 2013:
   http://www.mobileiron.com
6. SANS Computer Forensics Website 2013:
   http://computer-forensics.sans.org/blog/2012/04/09/is-anti-virus-really-dead-a-real-world-simulation-created-for-forensic-data-yields-surprising-results
7. XDA Developers Website Accessed February 2013:
   http://forum.xda-developers.com/showthread.php?t=1496777
8. GSM Arena Website Accessed February 2013:
   http://www.gsmarena.com/samsung_google_nexus_s-3620.php