

# Cellular Networks and Mobile Computing

COMS 6998-8, Spring 2012

Instructor: Li Erran Li  
([lierranli@cs.columbia.edu](mailto:lierranli@cs.columbia.edu))

<http://www.cs.columbia.edu/~coms6998-8/>

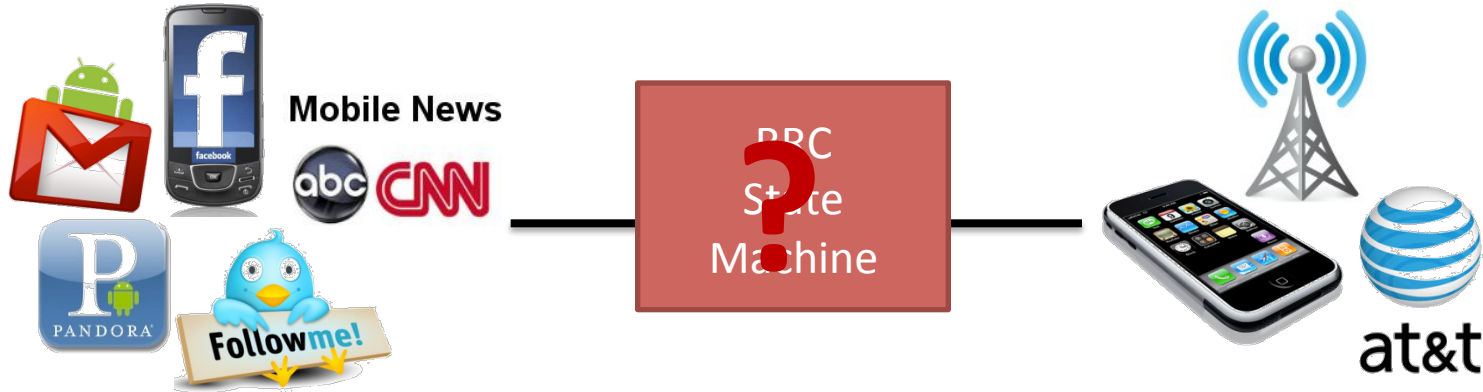
2/27/2012: Radio Resource Usage Profiling  
and Optimization

# Outline

- Introduction
- RRC State Inference
- Radio Resource Usage Profiling & Optimization
- Network RRC Parameters Optimization
- Conclusion

# Introduction

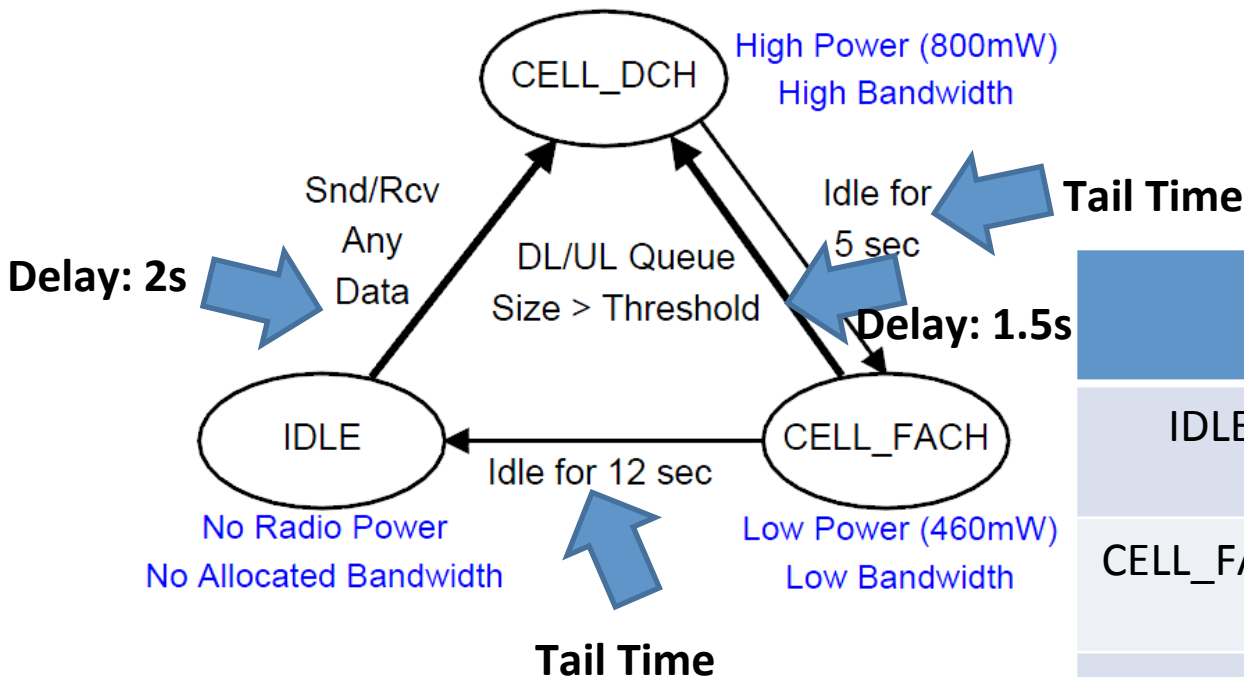
- Typical testing and optimization in cellular data network



- Little focus has been put on their **cross-layer interactions**  
Many mobile applications are not cellular-friendly.
- The key coupling factor: the **RRC State Machine**
  - Application traffic patterns trigger state transitions
  - State transitions control radio resource utilization, end-user experience and device energy consumption (battery life)

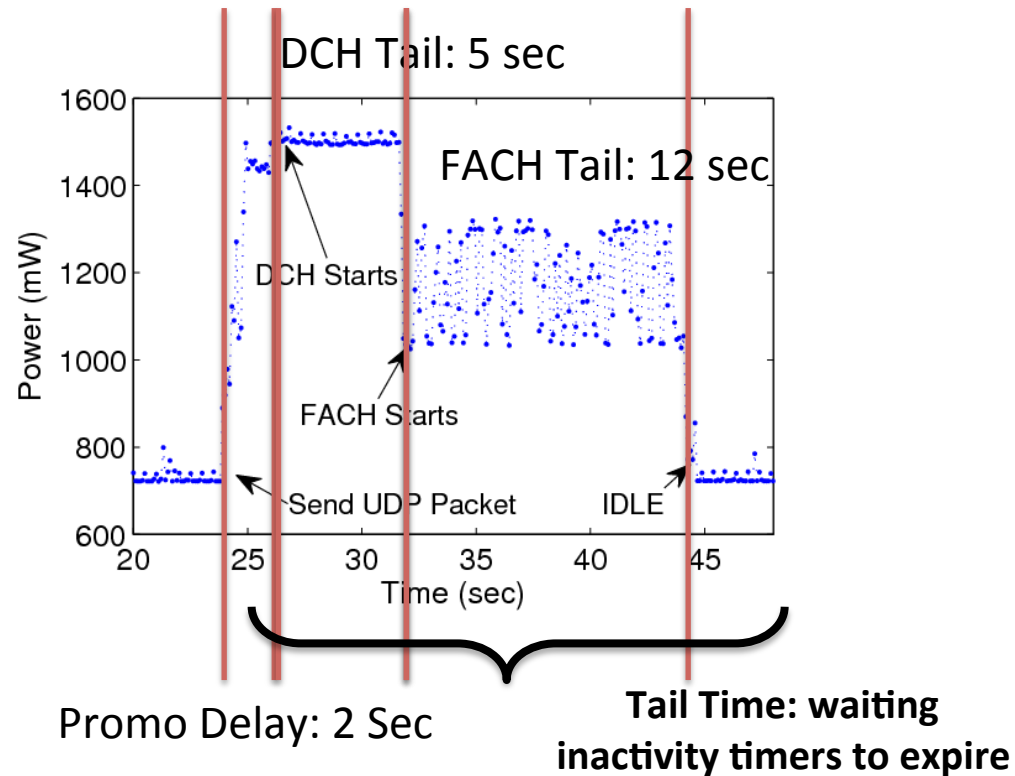
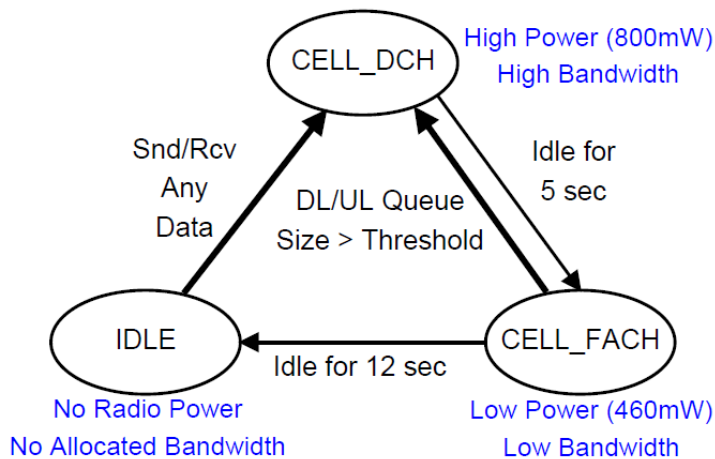
# The RRC State Machine for UMTS Network

- State promotions have **promotion delay**
- State demotions incur **tail times**



	Channel	Radio Power
IDLE	Not allocated	Almost zero
CELL_FACH	Shared, Low Speed	Low
CELL_DCH	Dedicated, High Speed	High

# Example: RRC State Machine for a Large Commercial 3G Network



- DCH:** High Power State (high throughput and power consumption)
- FACH:** Low Power State (low throughput and power consumption)
- IDLE:** No radio resource allocated

# Why State Promotion Slow?

- Tens of control messages are exchanged during a state promotion.

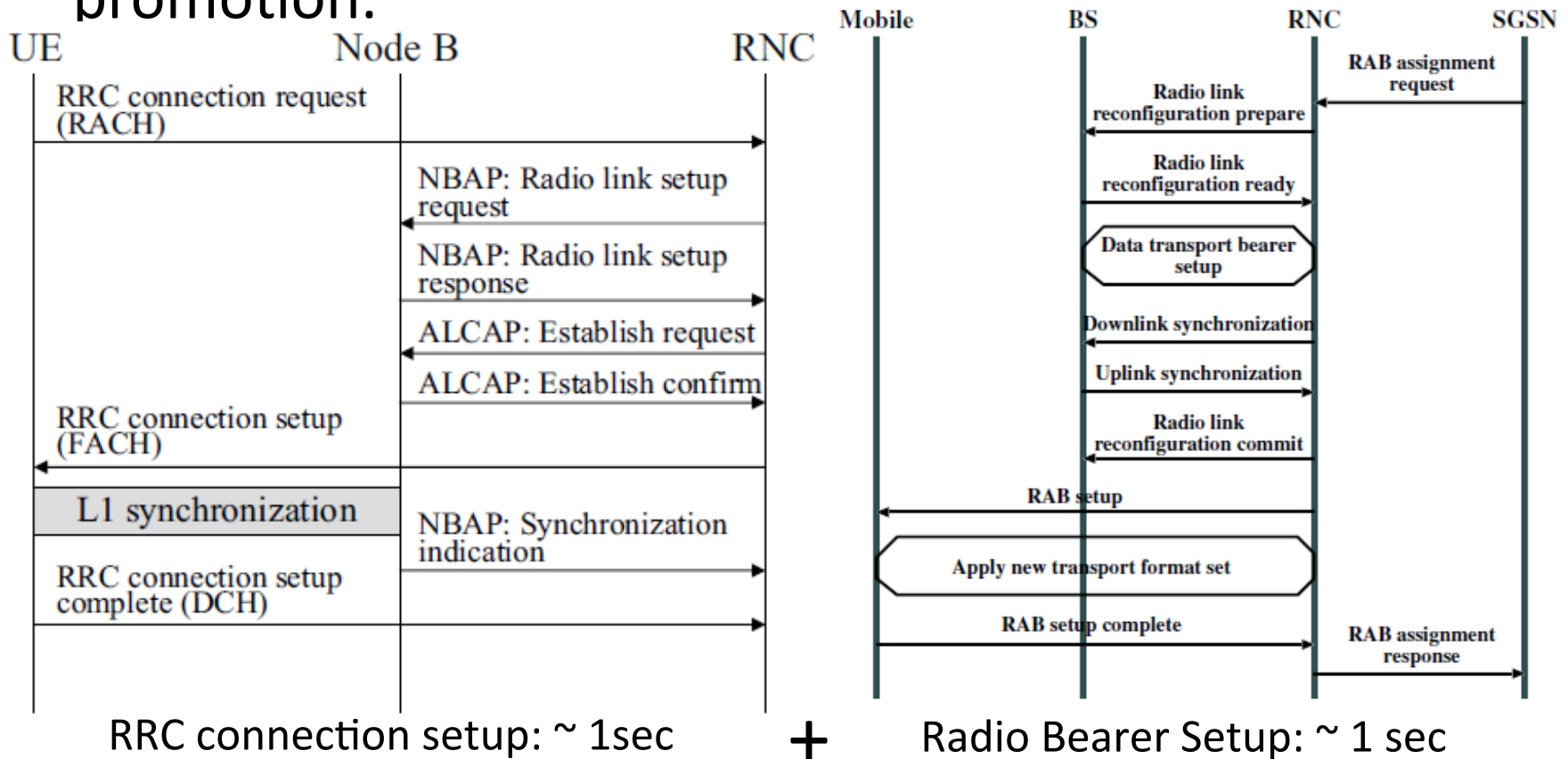
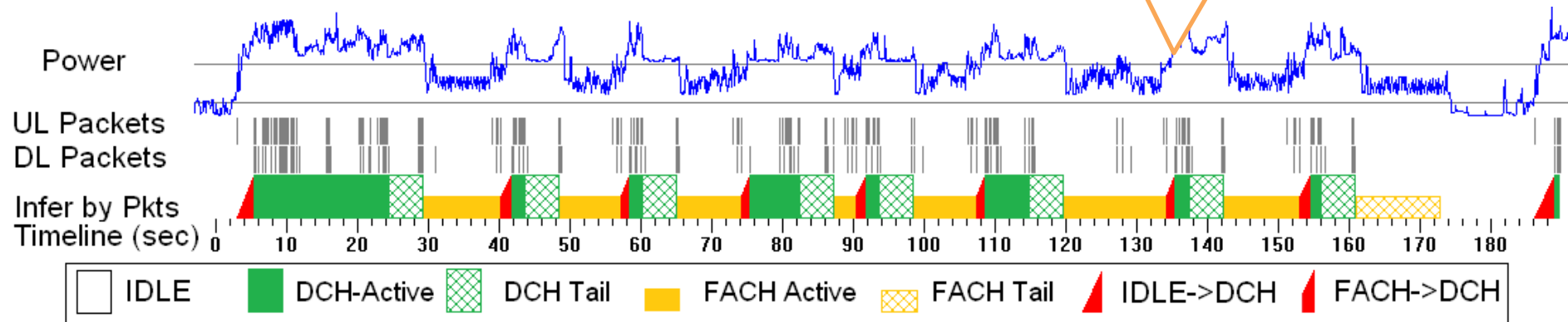


Figure source: HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications. *John Wiley and Sons, Inc., 2006.*

# Example of the State Machine Impact: Inefficient Resource Utilization

A significant amount of channel occupation time and battery life is wasted by **scattered bursts**.

State transitions impact end user experience and generate signaling load.



Analysis powered by the ARO tool

	FACH and DCH
<b>Wasted</b> Radio Energy	34%
<b>Wasted</b> Channel Occupation Time	33%

# Example of the State Machine Impact: DNS timeout in UMTS networks

Start from CELL\_DCH STATE (1 request / response) – Keep in DCH

240	11.806360	10.0.192.152	172.18.145.103	DNS	Standard query A www.eecs.umich.edu
241	11.991680	172.18.145.103	10.0.192.152	DNS	Standard query response A 141.212.113.110

Start from CELL\_FACH STATE (1 request / response) – Keep in FACH

237	18.304116	10.0.192.152	172.18.145.103	DNS	Standard query A www.eecs.umich.edu
238	18.627700	172.18.145.103	10.0.192.152	DNS	Standard query response A 141.212.113.110

Start from IDLE STATE (2~3 requests / responses) – IDLE → DCH

1884	221.958559	10.0.192.152	172.18.145.103	DNS	Standard query A www.eecs.umich.edu
1885	222.947858	10.0.192.152	172.18.145.103	DNS	Standard query A www.eecs.umich.edu
1886	223.947892	10.0.192.152	172.18.145.103	DNS	Standard query A www.eecs.umich.edu
1887	224.069453	172.18.145.103	10.0.192.152	DNS	Standard query response A 141.212.113.110
1888	224.070465	172.18.145.103	10.0.192.152	DNS	Standard query response A 141.212.113.110
1889	224.079463	172.18.145.103	10.0.192.152	DNS	Standard query response A 141.212.113.110

Starting from IDLE triggers at least one DNS timeout (default is 1 sec in WinXP)

2 second promotion delay because of the wireless state machine (see previous slide), but DNS timeout is 1 second!

**=> Triple the volume of DNS requests...**






# State Machine Inference

- State Promotion Inference
  - Determine one of the two promotion procedures
  - P1: IDLE → FACH → DCH;      P2: IDLE → DCH

---

## Algorithm 1 State promotion inference

---

- 1: Keep UE on IDLE.
  - 2: UE sends *min* bytes. Server echoes *min* bytes.  P1: IDLE → FACH, P2: IDLE → DCH
  - 3: UE sends *max* bytes. Server echoes *min* bytes.  P1: FACH → DCH, P2: Keep on DCH
  - 4: UE records the RTT  $\Delta t$  for Step 3.  Normal RTT < 300ms
  - 5: Report *P1* iff  $\Delta t \gg$  normal RTT. Otherwise report *P2*. RTT w/ Promo > 1500ms
- 

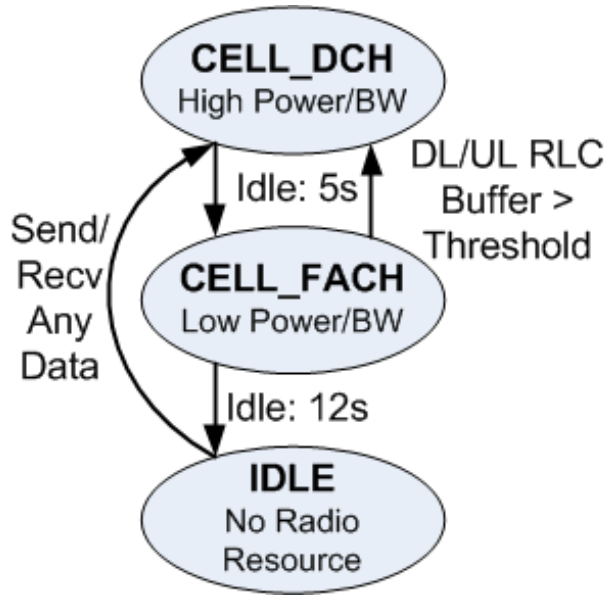
A packet of **min** bytes **never** triggers FACH → DCH promotion (we use 28B)

A packet of **max** bytes **always** triggers FACH → DCH promotion (we use 1KB)

- State demotion and inactivity timer inference
  - See paper for details

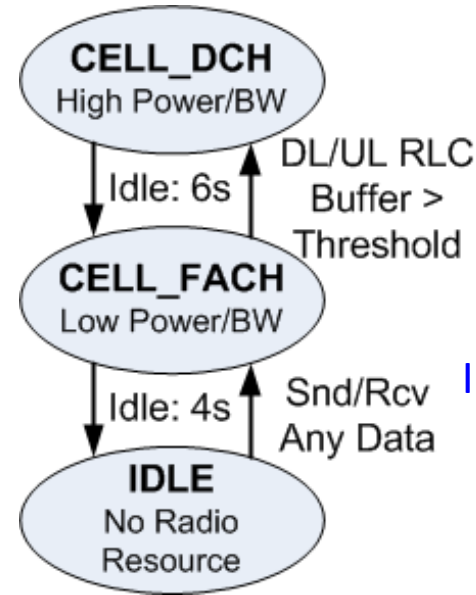
# RRC State Machines of Two Commercial UMTS Carriers

Promotion  
Inference  
Reports **P2**  
IDLE → DCH



Carrier 1

Promotion  
Inference  
Reports **P1**  
IDLE → FACH → DCH



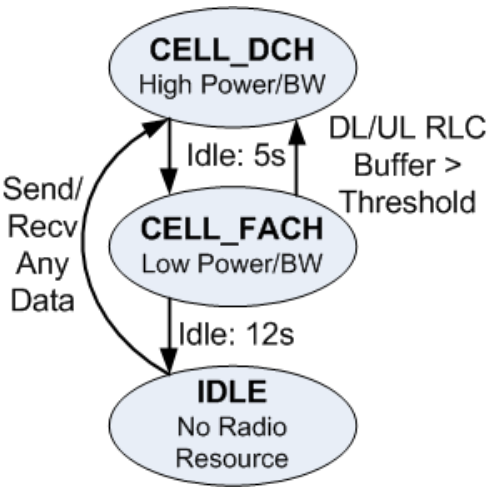
Carrier 2

Timer	Carrier 1	Carrier 2
DCH → FACH ( $\alpha$ timer)	5 sec	6 sec
FACH → IDLE ( $\beta$ timer)	12 sec	4 sec

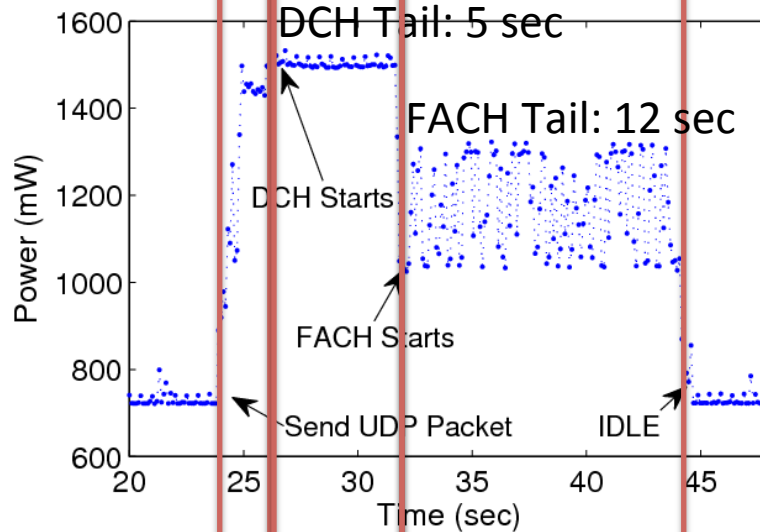
**What are the optimal inactivity timer values?**

# State Machine Inference

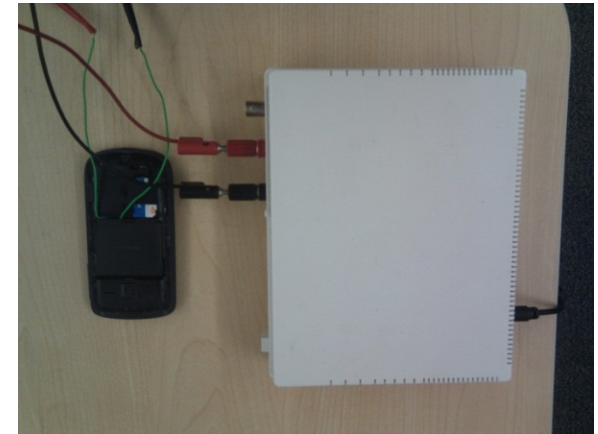
- Validation using a power meter



Carrier 1



Promo Delay: 2 Sec



RRC State	Avg Radio Power
IDLE	0
FACH	460 mW
DCH	800 mW
FACH → DCH	700 mW
IDLE → DCH	550 mW

# Outline

- Introduction
- RRC State Inference
- Radio Resource Usage Profiling & Optimization
- Network RRC Parameters Optimization
- Conclusion

# ARO: Mobile **A**pplication **R**esource **O**ptimizer

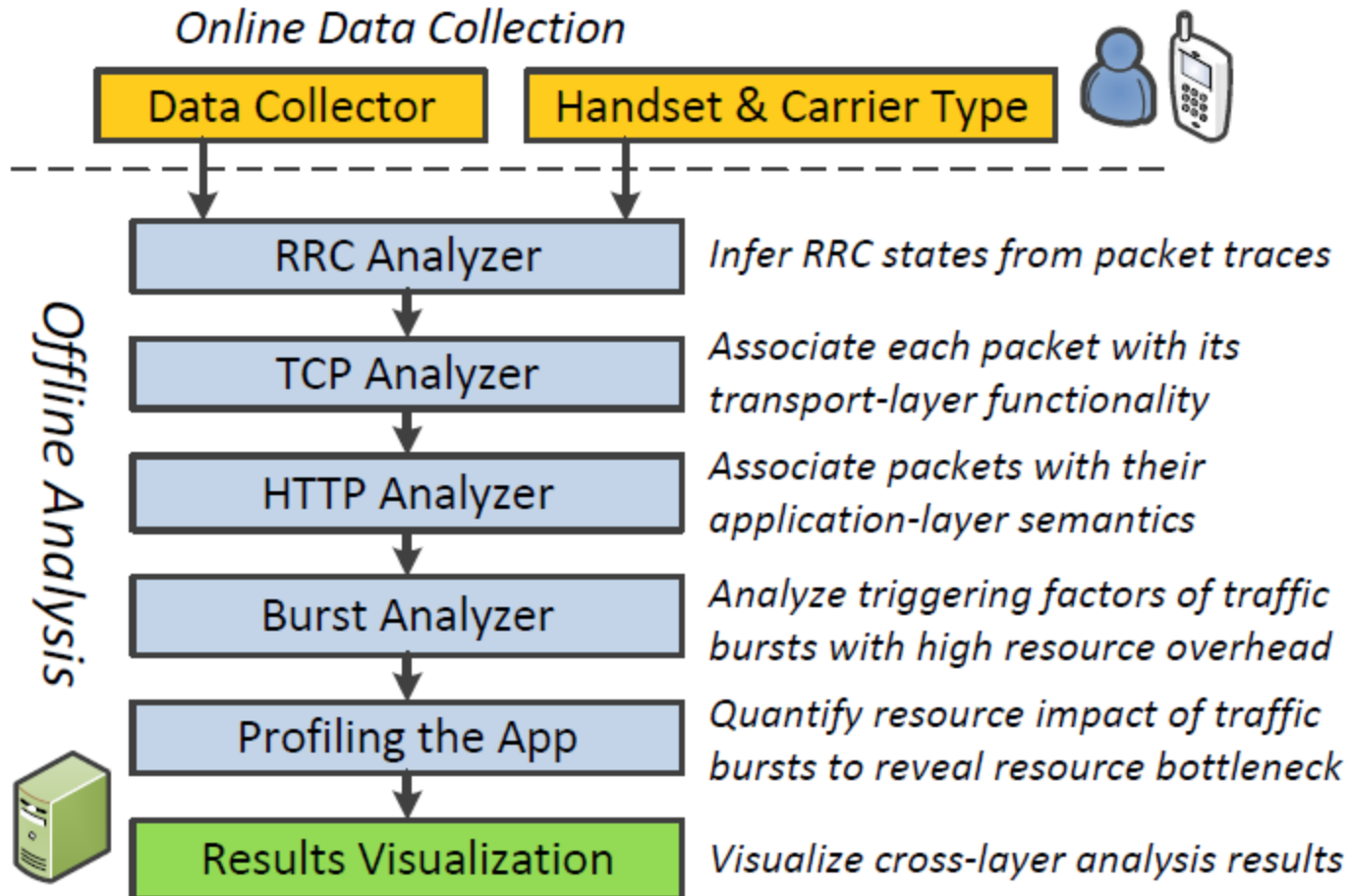
- **Motivations:**

- Are developers aware of the RRC state machine and its implications on radio resource / energy? **NO.**
- Do they need a tool for automatically profiling their prototype applications? **YES.**
- If we provide that visibility, would developers optimize their applications and reduce the network impact? **Hopefully YES.**

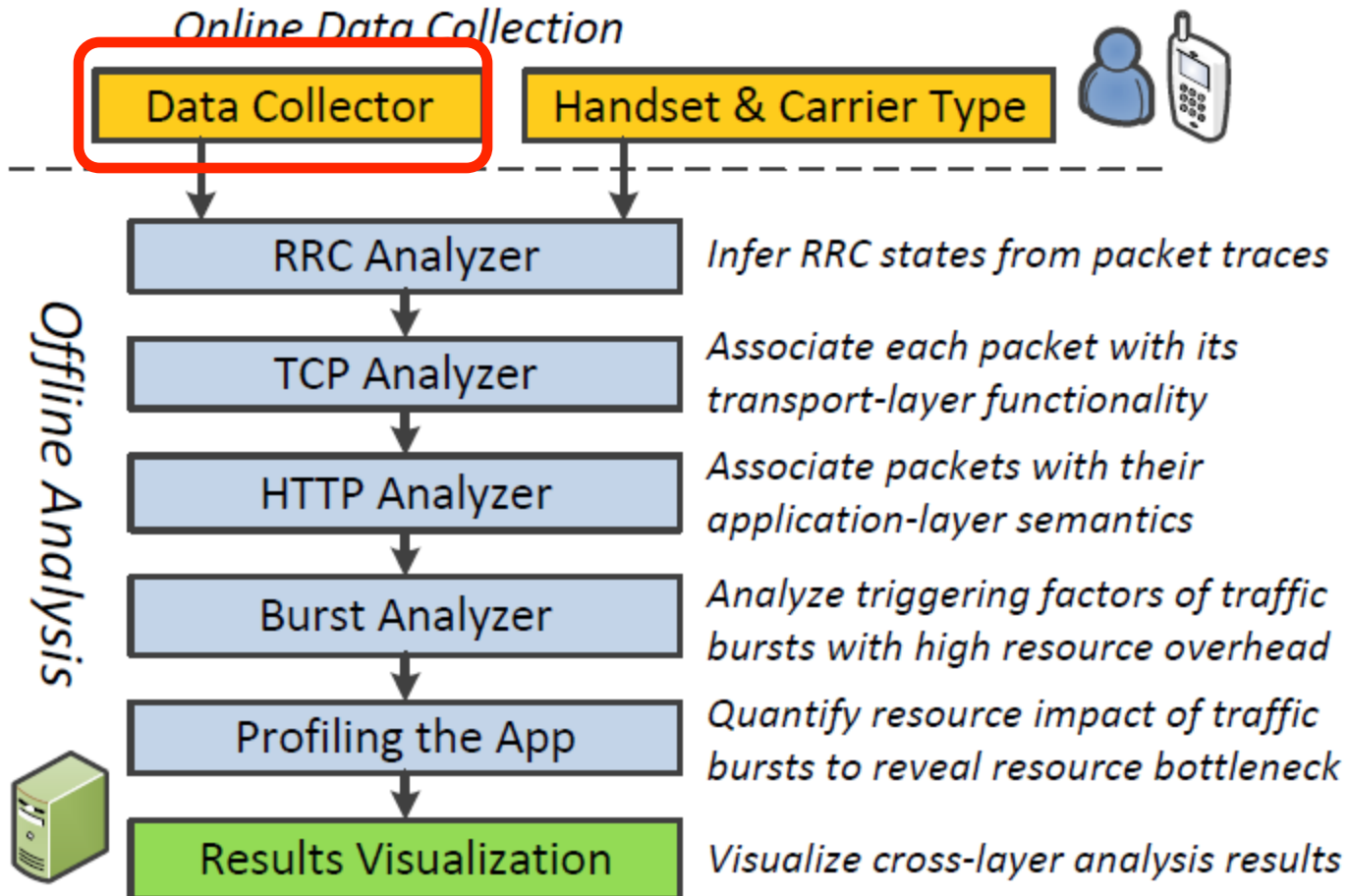
- **ARO: Mobile Application Resource Optimizer**

- Provide visibility of radio resource and energy utilization.
- Benchmark efficiencies of cellular radio resource and battery life for a specific application

# ARO System Architecture



# ARO System Architecture

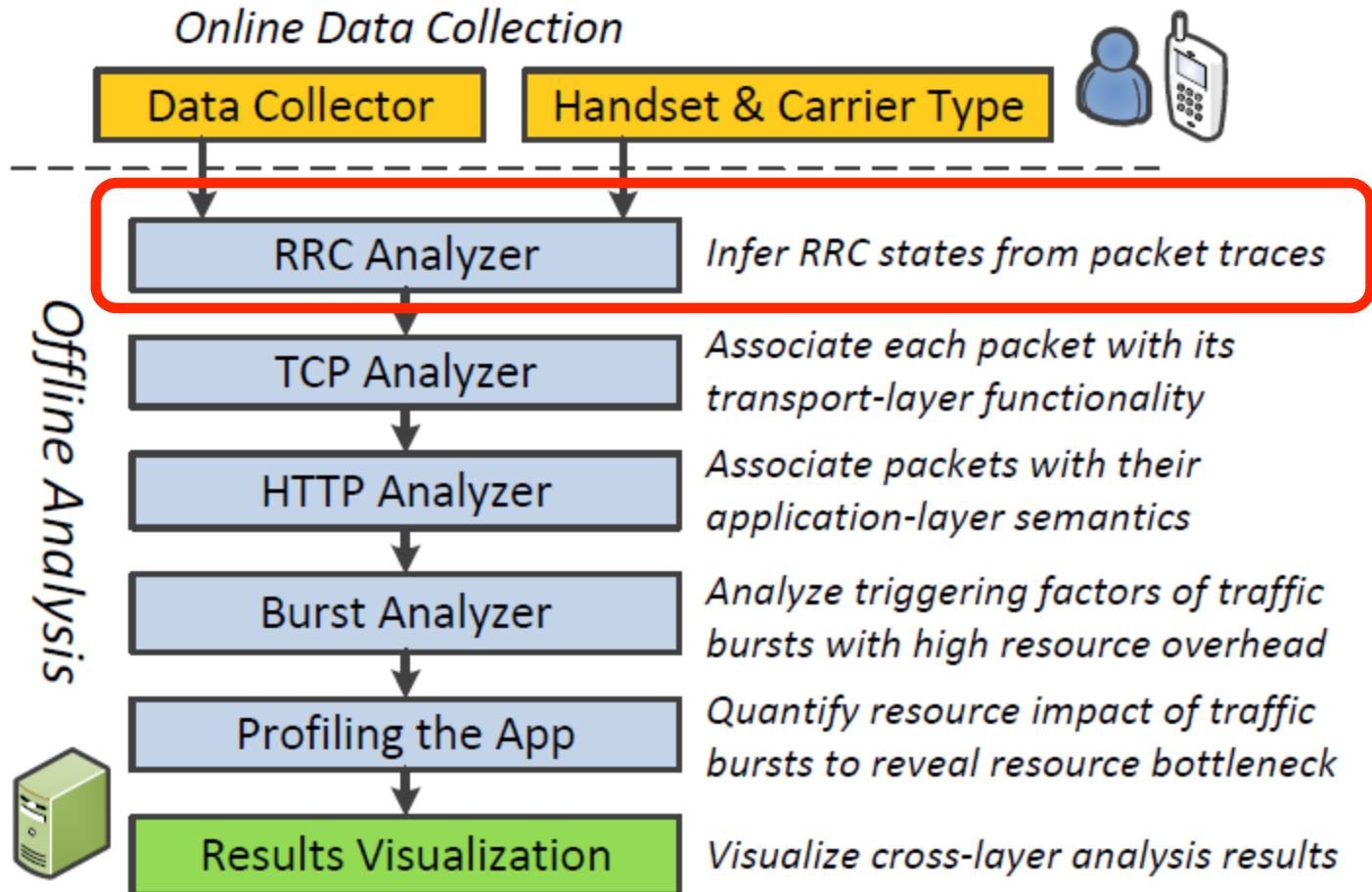


# The Data Collector

- Collects three pieces of information
  - The packet trace
  - User input (e.g., touching the screen)
  - Packet-process correspondence
    - The RRC state transition is triggered by the **aggregated** traffic of all concurrent applications
    - But we are only interested in our **target** application.
- Less than 15% runtime overhead when the throughput is as high as 600kbps

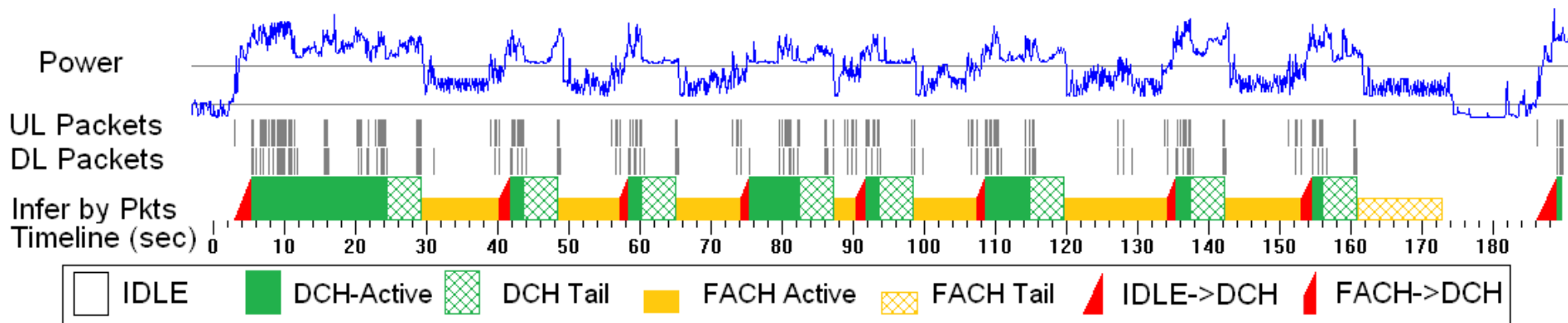


# ARO System Architecture



# RRC Analyzer: State Inference

- RRC state inference
  - Taking the packet trace as input, **simulate** the RRC state machine to infer the RRC states
    - Iterative packet driven simulation: given RRC state known for  $\text{pkt}_i$ , infer state for  $\text{pkt}_{i+1}$  based on inter-arrival time, packet size and UL/DL
  - Evaluated by measuring the device power



## Example: Web Browsing Traffic on HTC TyTn II Smartphone

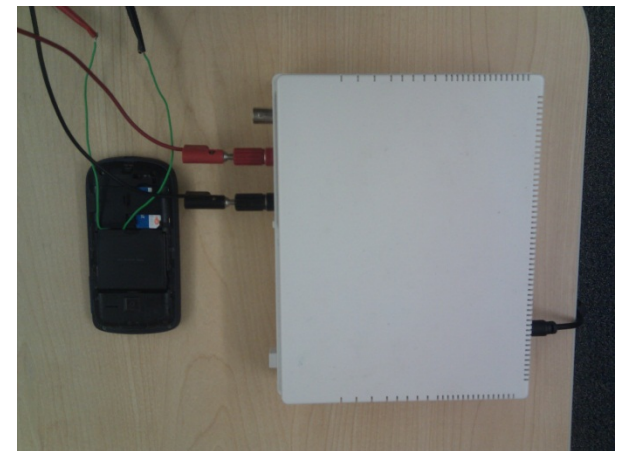
# RRC Analyzer: Applying the Energy Model

- Apply the energy model
  - Associate each state with a constant power value
  - Based on our measurement using a power-meter

Table 3: Measured average radio power consumption

	TyTn Carrier 1	NexusOne Carrier 1	ADP1 * T-Mobile
$P(\text{IDLE})$	0	0	10mW
$P(\text{FACH})$	460mW	450mW	401mW
$P(\text{DCH})$	800mW	600mW	570mW
$P(\text{FACH} \rightarrow \text{DCH})$	700mW	550mW	N/A
$P(\text{IDLE} \rightarrow \text{DCH})$	550mW	530mW	N/A

\* Reported by [27] for Android HTC Dream phone



# RRC Analyzer: Applying the Energy Model (Cont'd)

- 3G radio interface power consumption
  - at DCH, the radio power (800 mW) contributes 1/3 to 1/2 of total device power (1600 mW to 2400 mW)

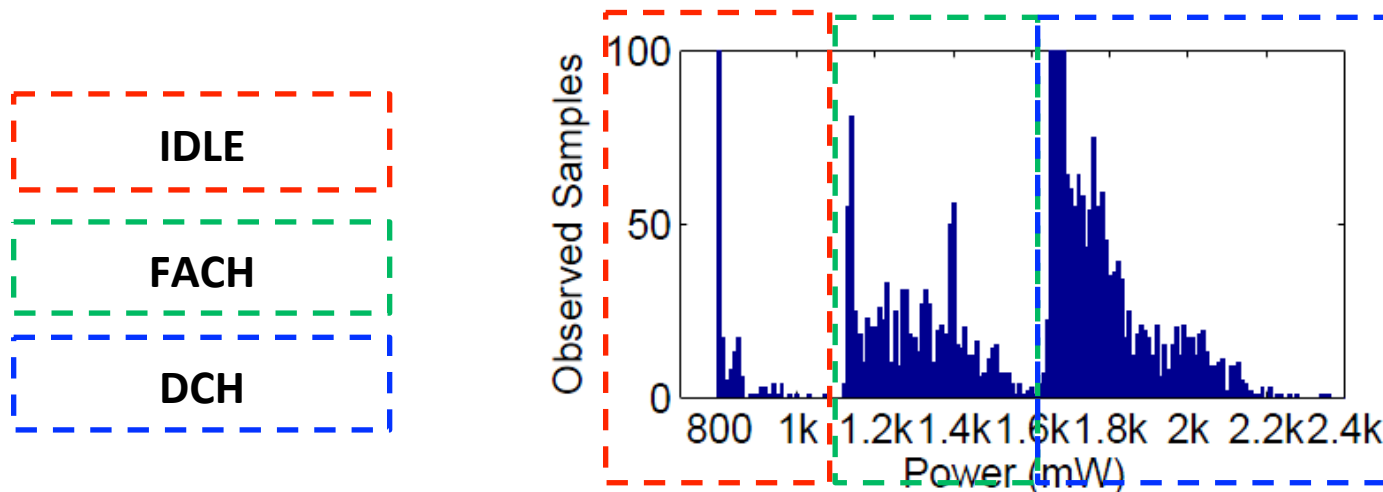
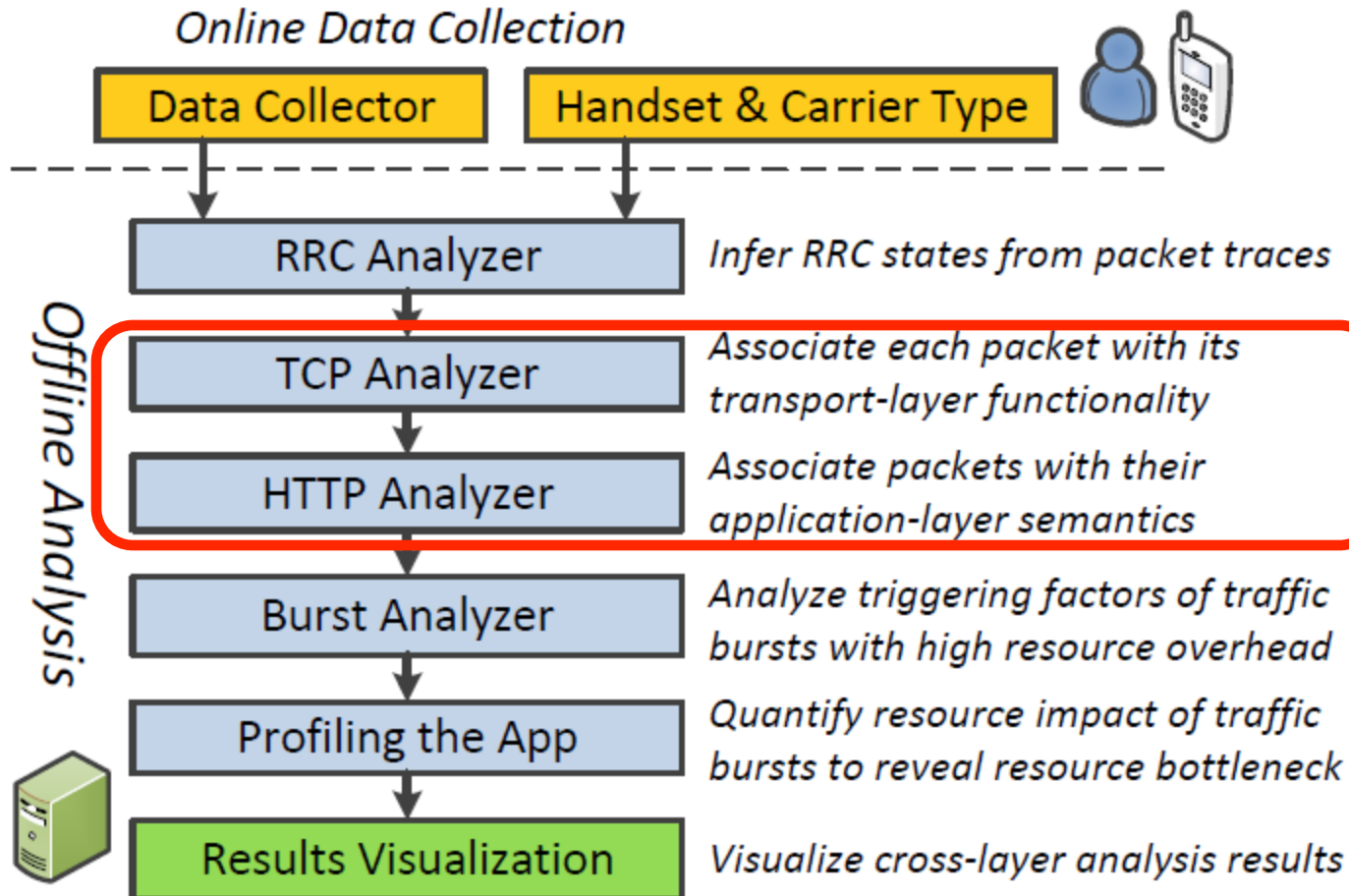


Figure 8: Histogram of measured power values for the News1 trace collected at an HTC TyTn II phone

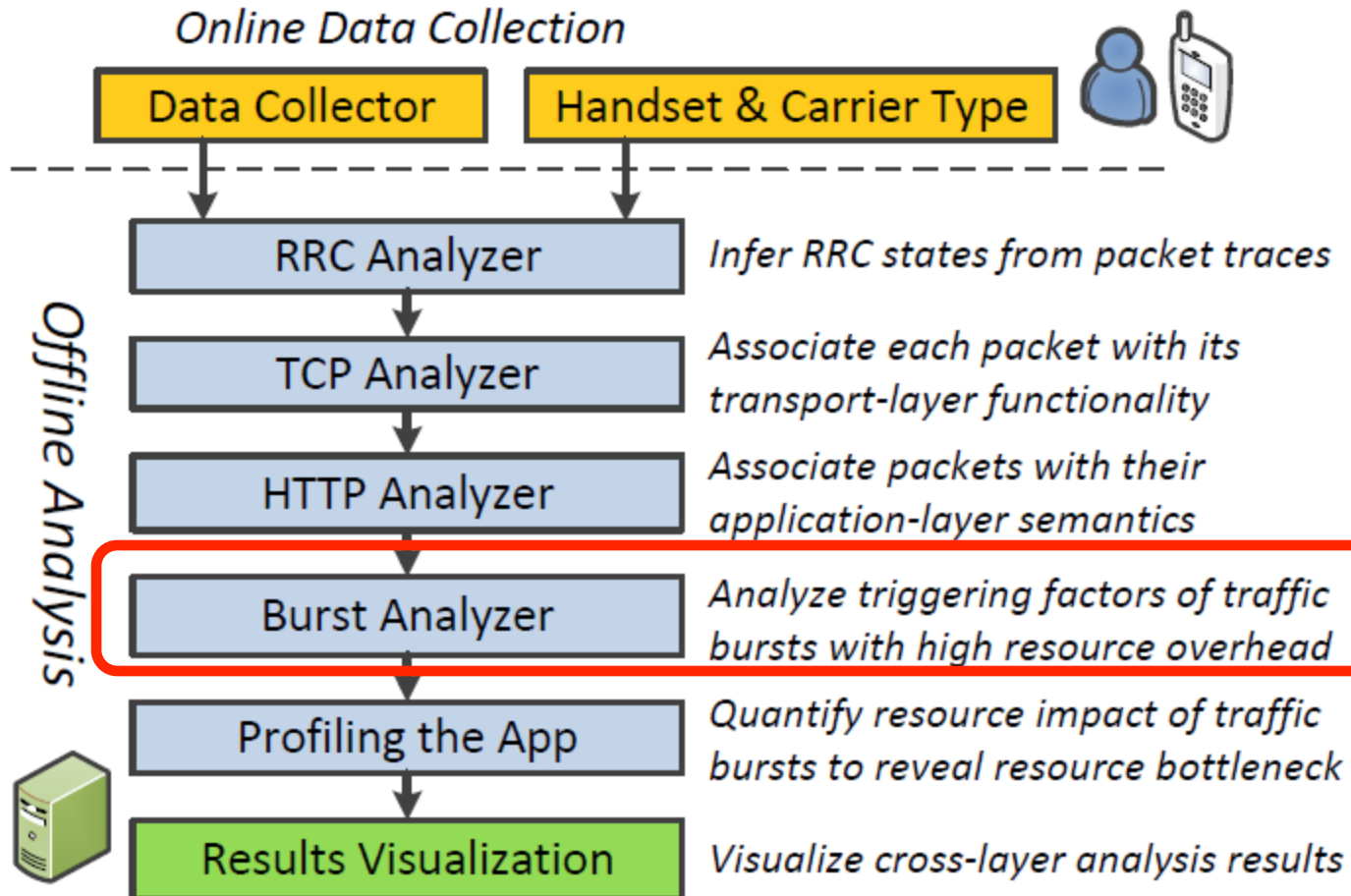
# ARO System Architecture



# TCP / HTTP Analysis

- TCP Analysis
  - Infer transport-layer properties for each TCP packet
    - SYN, FIN, or RESET?
    - Related to loss? (e.g., duplicated ACK / recovery ACK)
    - ...
- HTTP Analysis:
  - HTTP is the dominant app-layer protocol for mobile apps.
  - Model HTTP behaviors

# ARO System Architecture



# Burst Analysis

- A burst consists of consecutive packets transferred in a batch (i.e., their IAT is less than a threshold)
- We are interested in **short bursts** that incur energy / radio resource inefficiencies
- ARO finds the **triggering factor** of each short burst
  - Triggered by user interaction?
  - By server / network delay?
  - By application delay?
  - By TCP protocol?





# Burst Analysis Algorithm

```

01 Burst_Analysis (Burst  $b$ ) {
02   Remove packets of non-target apps;
03   if (no packet left) {return NON_TARGET;} Test 1
04   if ( $b.payload > th_s$  &&  $b.duration > th_d$ ) Test 2
05     {return LARGE_BURST;}
06   if ( $b.payload == 0$ ) { Test 3
07     if ( $b$  contains any of ESTABLISH, CLOSE, RESET,
08     TCP_OTHER packets)
09     {return TCP_CONTROL;}
10 }

```

```

11  $d_0 \leftarrow$  direction of the first packet of  $b$ ;
12  $i_0 \leftarrow$  TCP label of the first packet of  $b$ ;
13 if ( $d_0 == DL$  && ( $i_0 == DATA$  ||  $i_0 == ACK$ )) Test 4
14   {return SVR_NET_DELAY;}
15 if ( $i_0 == ACK_DUP$  &&  $i_0 == ACK_RECOVER$  &&
16  $i_0 == DATA_DUP$  &&  $i_0 == DATA_RECOVER$ ) Test 5
17   {return TCP_LOSS_RECOVER;}
18 if ( $b.payload > 0$  && find user input before  $b$ ) Test 6
19   {return USER_INPUT;}
20 if ( $b.payload > 0$ ) {return APP;} Test 7
21 else {return UNKNOWN;}
22 }

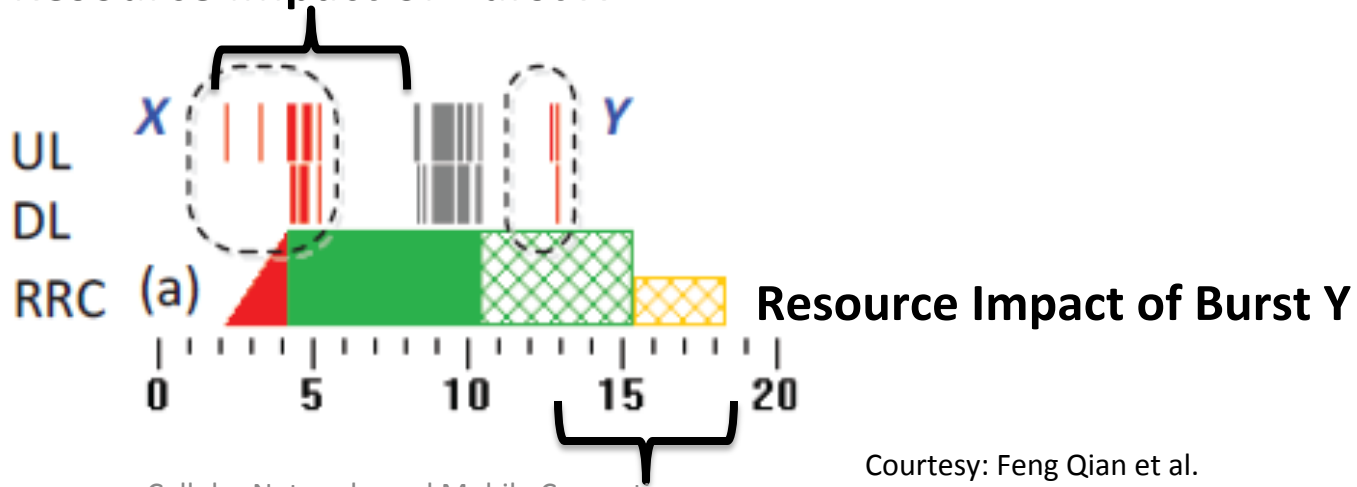
```

Label	The burst is triggered by ...
USER_INPUT	User interaction
LARGE_BURST	(The large burst is resource efficient)
TCP_CONTROL	TCP control packets ( <i>e.g.</i> , FIN and RST)
SVR_NET_DELAY	Server or network delay
TCP_LOSS_RECOVER	TCP congestion / loss control
NON_TARGET	Other applications not to be profiled
APP	The application itself
APP_PERIOD	Periodic data transfers (One special type of APP)

# Compute Resource Consumption of a Burst

- **Upperbound** of resource utilization
  - The resource impact of a burst  $B_i$  is from the beginning of  $B_i$  to the beginning of the next burst  $B_{i+1}$
  - May **overestimate** resource consumption, as one burst may already be covered by the tail of the previous burst

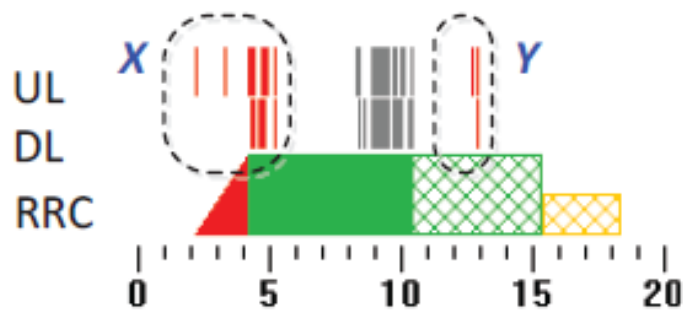
Resource Impact of Burst X



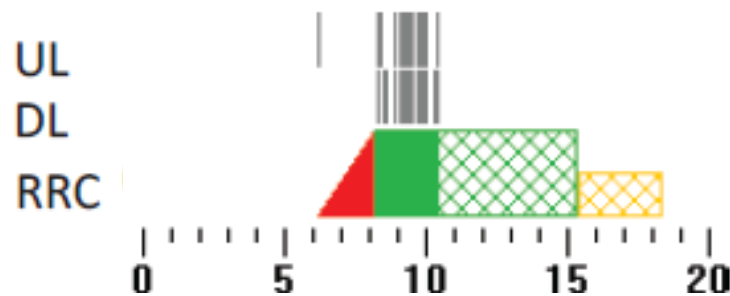
Courtesy: Feng Qian et al.

# Compute Resource Consumption of a Burst

- **Lowerbound** of resource utilization
  - Compute the total resource utilization of the original trace
  - Remove the interested burst, then compute the resource utilization again
  - Take the delta



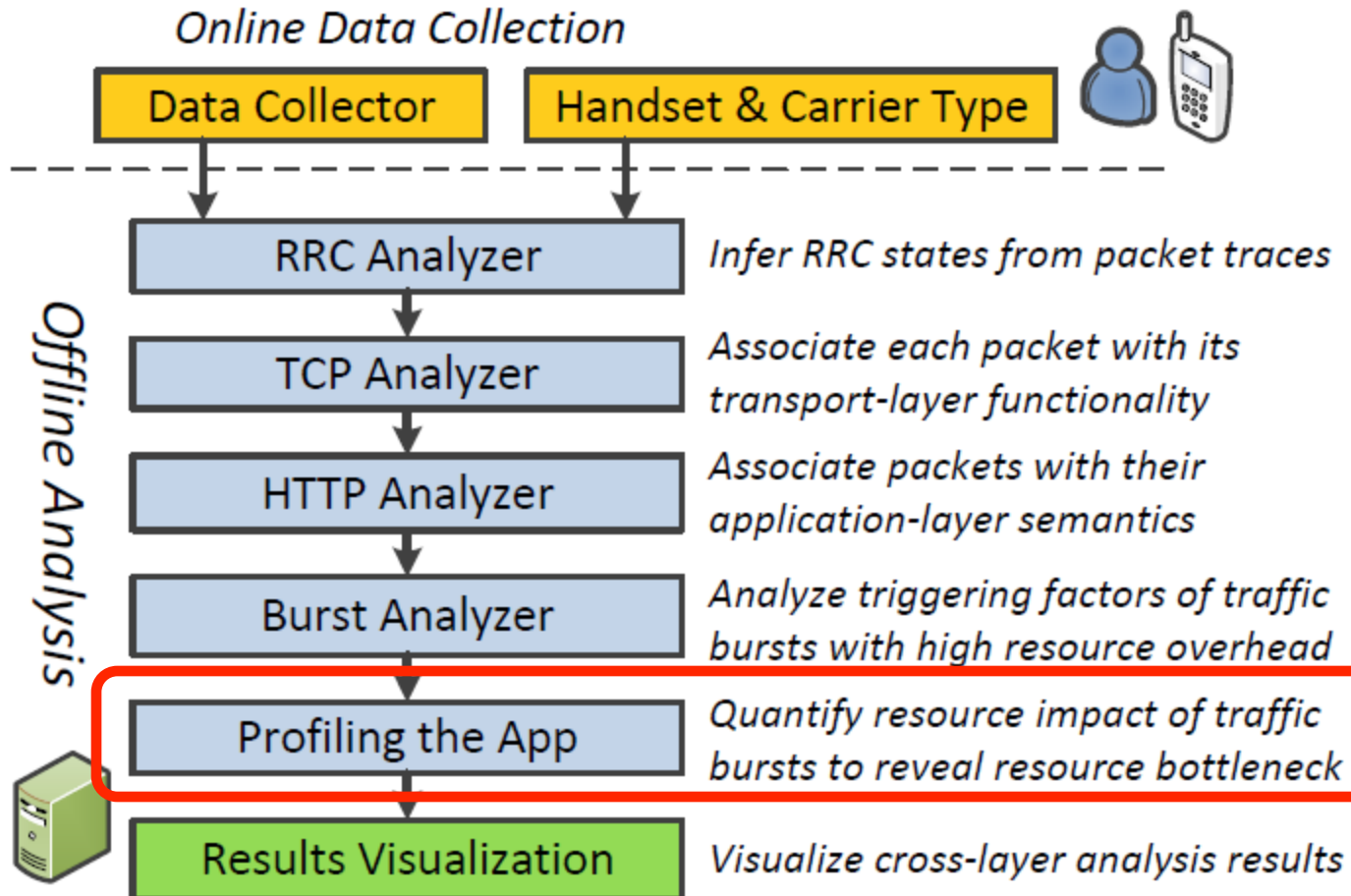
The original trace  
Resource Utilization is  $E_1$



Remove X and Y  
Resource utilization is  $E_2$

The resource impact of X and Y is  $E_1 - E_2$

# ARO System Architecture



# Profiling Applications

- From **RRC Analysis**
  - We know the **radio resource state** and the **radio power** at any given time
- From **Burst analysis**
  - We know the **triggering factor** of each burst
  - We know the **transport-layer** and **application-layer** behavior of each burst
- By “**profiling applications**”, we mean
  - Compute resource consumption of each **burst**
  - Therefore identify the root cause of resource inefficiency.

# Metrics for Quantifying Resource Utilization Efficiency

- Handset radio energy consumption
- DCH occupation time
  - Quantifies **radio resource utilization**
- Total state promotion time (IDLE → DCH, FACH → DCH)
  - Quantifies **signaling overhead**
- Details of computing the three metrics (upperbound and lowerbound) in the paper

# Implementation

- Data collector built on Android: modified tcpdump with two new features (1K lines of code)
  - logging user inputs: reads `/dev/input/event*`
    - captures all user input events such as touching the screen, pressing buttons
  - finding packet-to-application association
    - `/proc/PID/fd` containing mappings from process ID (PID) to inode of each TCP/UDP socket
    - `/proc/net/tcp(udp)` maintaining socket to inode mappings,
    - `/proc/PID/cmdline` that has the process name of each PID
- The analyzers were implemented in C++ on Windows 7 (7.5K lines of code)

# Case Studies

- Fully implemented for Android platform (7K LoC)
- Study 17 popular Android applications
  - All in the “TOP Free” Section of Android Market
  - Each has 250,000+ downloads as of Dec 2010
- ARO pinpoints resource inefficiency for many popular applications. For example,
  - **Pandora Streaming**  
High radio energy overhead (50%) of periodic measurements
  - **Fox News**  
High radio energy overhead (15%) due to users’ scrolling
  - **Google Search**  
High radio energy overhead (78%) due to real-time query suggestions

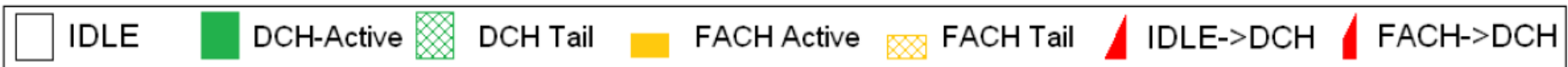
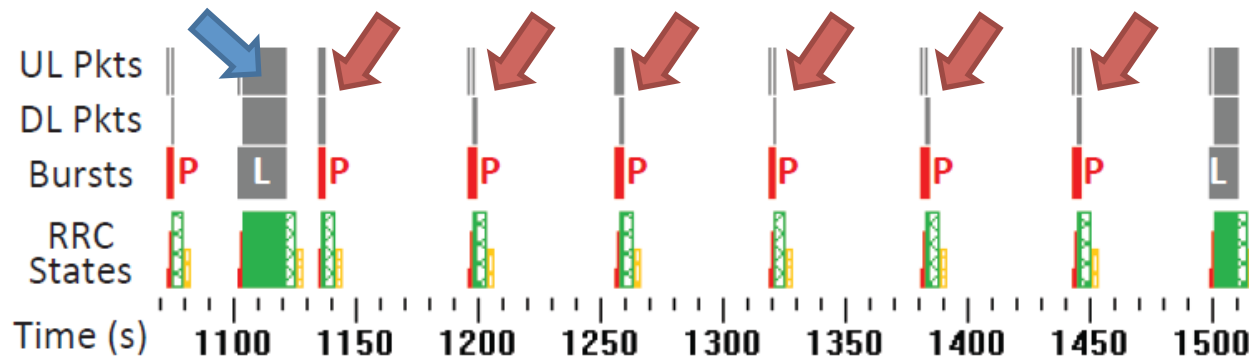


# Case Study: Pandora Music

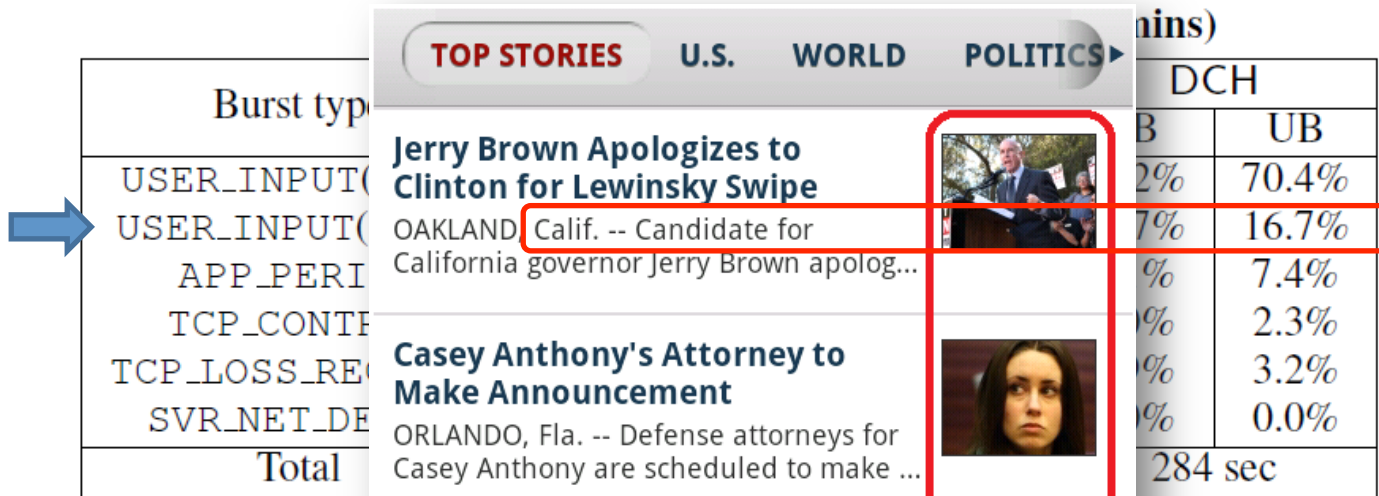
Pandora profiling results (Trace len: 1.45 hours)

Burst type	Payloads	Energy		DCH	
		LB	UB	LB	UB
LARGE_BURST	96.4%	35.6%	35.9%	42.4%	42.5%
APP_PERIOD	0.2%	45.9%	46.7%	40.4%	40.9%
APP	3.2%	12.8%	13.4%	12.4%	12.8%
TCP_CONTROL	0.0%	1.2%	1.6%	1.1%	1.5%
TCP_LOSS_RECOVER	0.2%	0.2%	0.6%	0.3%	0.7%
NON_TARGET	0.0%	1.8%	1.8%	1.7%	1.7%
Total	23.6 MB	846 J		895 sec	

**Problem:** High resource overhead of periodic audience measurements (every 1 min)  
**Recommendation:** Delay transfers and batch them with delay-sensitive transfers

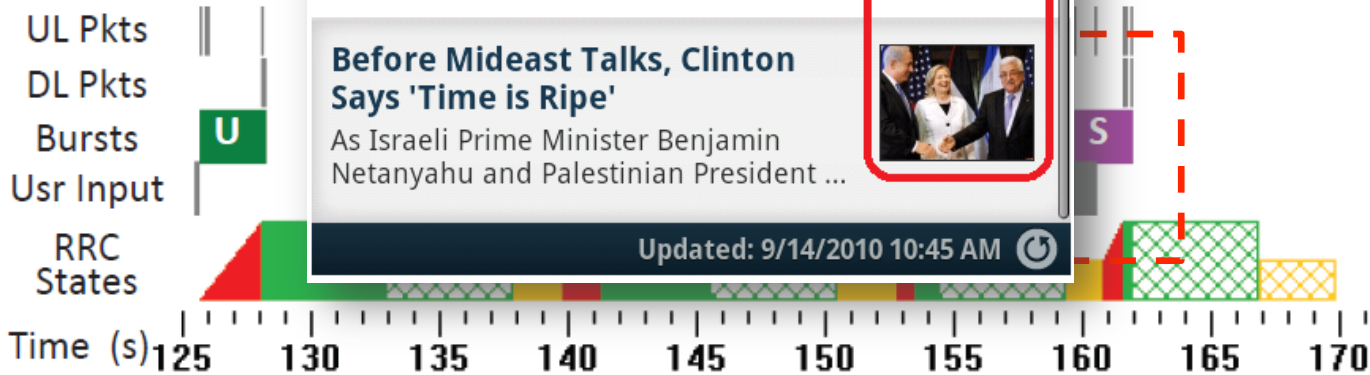


# Case Study: Fox News



**Problem:** Scattered burst  
**Recommendation:** G

Images in one burst



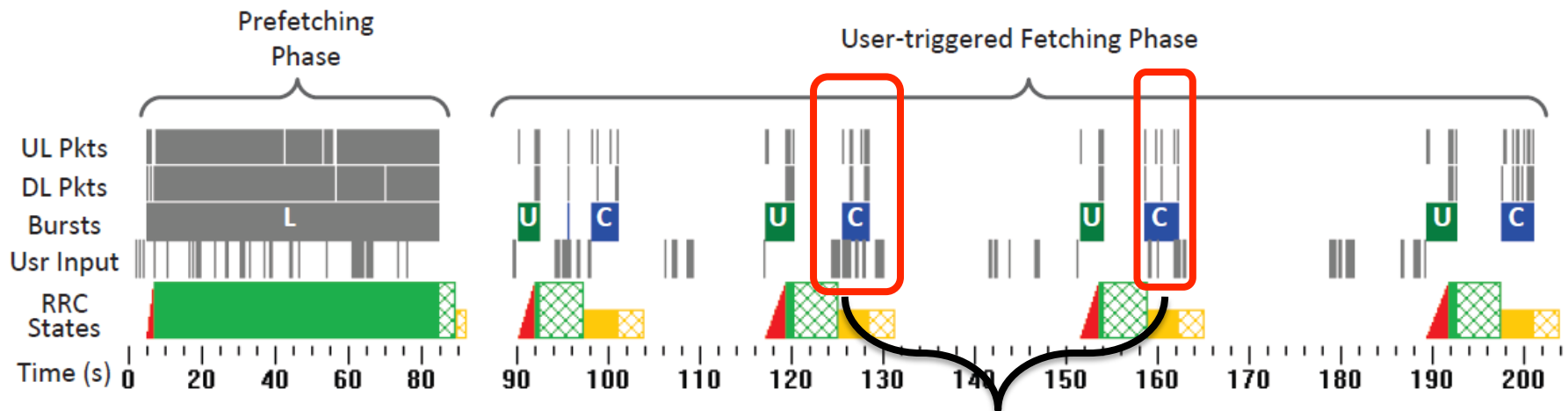
# Case Study: BBC News

## BBC News profiling results

User-triggered Fetching Phase (8 mins)					
Burst type	Payloads	Energy		DCH	
		LB	UB	LB	UB
TCP_CONTROL	0	11.3%	24.2%	0.0%	5.7%
USER_INPUT	98.7%	42.5%	73.1%	37.9%	90.0%
SVR_NET_DELAY	1%	0.0%	2.7%	0.0%	5.2%
Total	162 KB	145 J		120 sec	

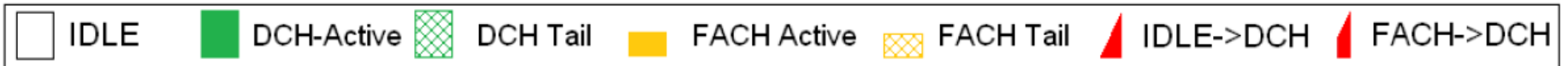
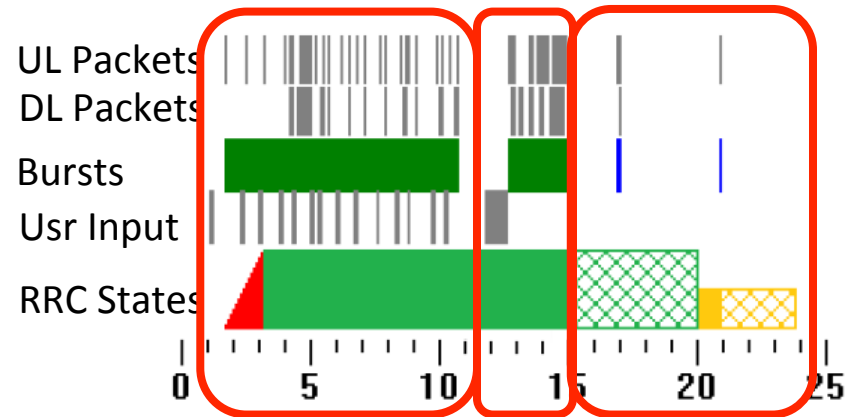
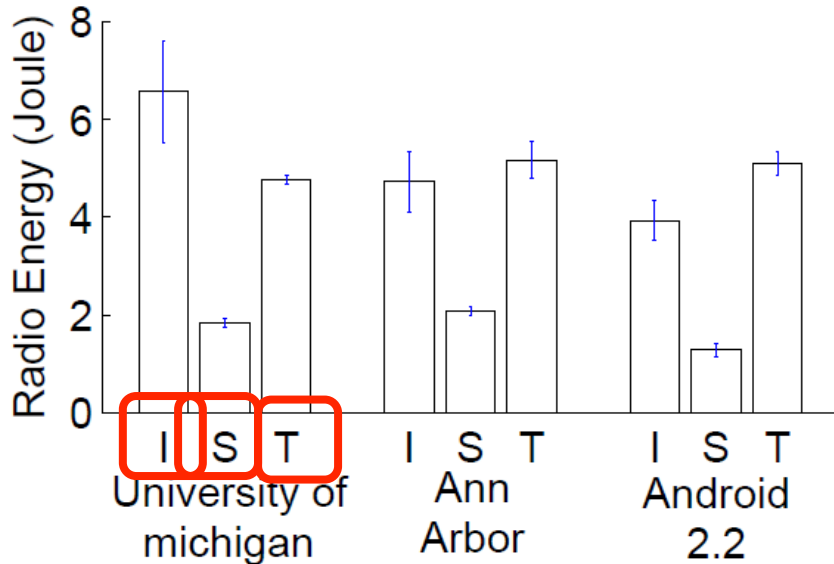
**Problem:** Scattered bursts of delayed FIN/RST packets

**Recommendation:** Close a connection immediately if possible, or within tail time



**Scattered bursts of delayed  
FIN/RST Packets**

# Case Study: Google Search



Search three key words.

ARO computes energy consumption for three phases

**I: Input phase S: Search phase T: Tail Phase**

**Problem:** High resource overhead of query suggestions and instant search

**Recommendation:** Balance between functionality and resource when battery is low

# Case Study: Audio Streaming

## Constant bitrate vs. bursty streaming

Name	Server	bitrate	Radio Power
NPR News	SHOUTcast	32 kbps	36 J/min
Tune-in	Icecast	119 kbps	36 J/min
Iheartradio	QTSS	32 kbps	36 J/min
Pandora w/ Ad	Apache	bursty	11.2 J/min
Pandora w/o Ad*	Apache	bursty	4.8 J/min
Slacker	Apache	bursty	10.9 J/min

\* A hypothetical case where all periodic ads are removed.

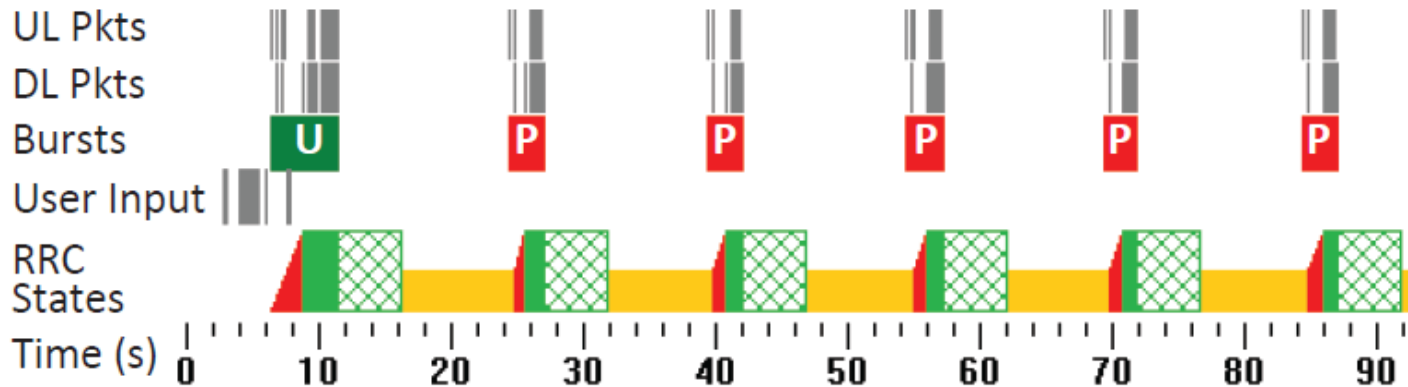
**Problem:** Low DCH utilization due to constant-bitrate streaming

**Recommendation:** Buffer data and periodically stream data in one burst

# Case Study: Mobile Advertisements

Comparing three mobile ad platforms

Name	Default Refresh Rate	Avg Up-date Size	Radio Power	
			w/ FD	w/o FD
Google Mobile Ad	180.0 sec	6.0 KB	2.5 J/min	3.6 J/min
AdMob	62.5 sec	6.8 KB	5.7 J/min	8.8 J/min
Mobclix	15.0 sec	1.4 KB	23.2 J/min	29.6 J/min



**Problem:** Aggressive ad refresh rate making the handset persistently occupy FACH or DCH

**Recommendation:** Decrease the refresh rate, piggyback or batch ad updates

# Outline

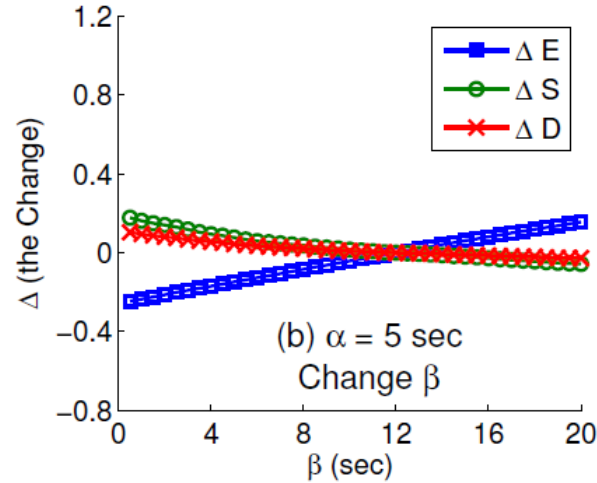
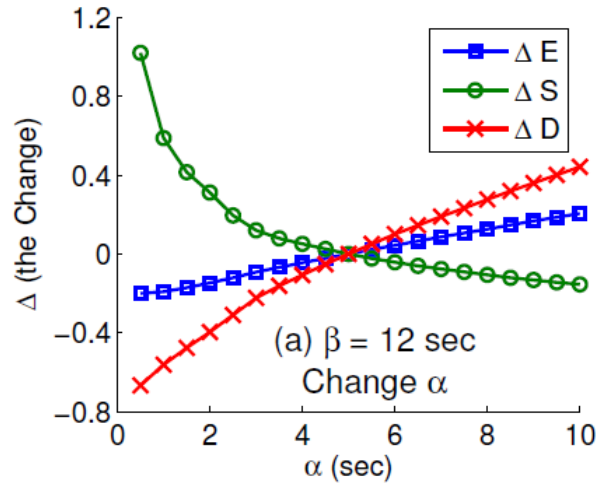
- Introduction
- RRC State Inference
- Radio Resource Usage Profiling & Optimization
- **Network RRC Parameters Optimization**
- **Conclusion**

# What-if Analysis for Inactivity Timers

- Inactivity timers are the most crucial parameters affecting
  - UE energy consumption
  - State promotion overhead
  - Radio resource utilization (i.e., DCH occupation time)
- What is the impact of changing inactivity timers
  - Perform **what-if analysis** by replaying traces to the simulator with different inactivity timer values.



# What-if Analysis for Inactivity Timers (Cont'd)



	Relative Change of...
$\Delta E$	Radio Energy
$\Delta S$	Promotion Delay
$\Delta D$	DCH Occupation Time

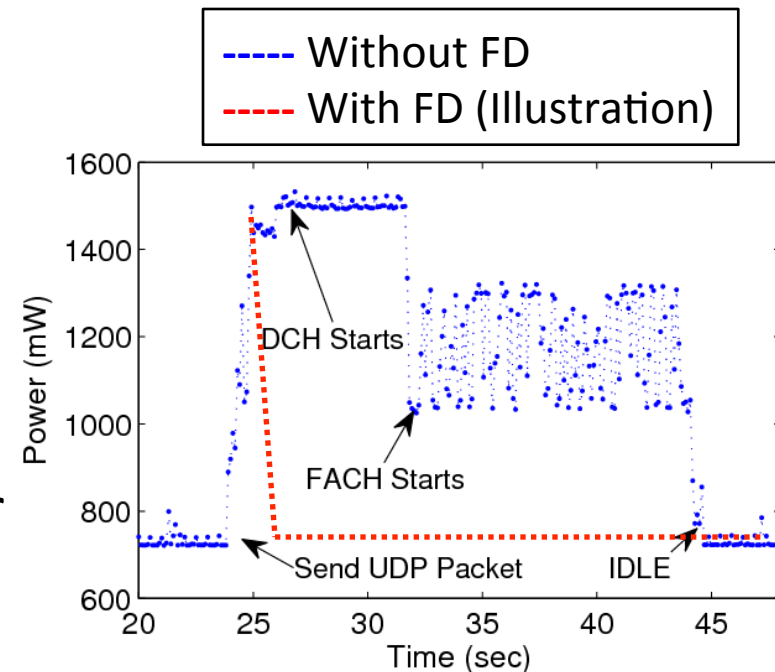
Fix the  $\alpha$  (DCH→FACH) timer  
Change the  $\beta$  (FACH→IDLE) timer

Fix the  $\beta$  (FACH→IDLE) timer  
Change the  $\alpha$  (DCH→FACH) timer

- The  $\alpha$  (DCH→FACH) timer imposes much higher impact on the three metrics than the  $\beta$  (FACH→IDLE) timer does.
- Very small  $\alpha$  timer values (< 2 sec) cause significant increase of state promotion overhead.
- **It is difficult to well balance the tradeoff. The fundamental reason is that timers are globally and statically set to constant values.**

# Fast Dormancy

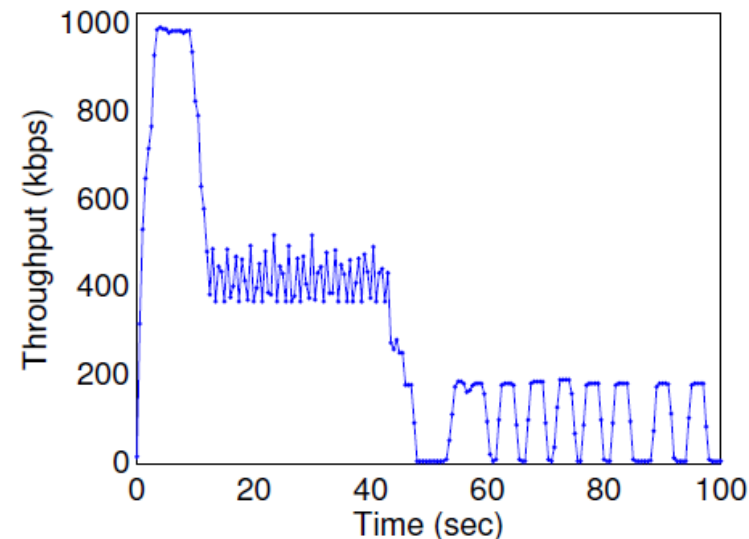
- A new feature added in 3GPP Release 7
- When finishing transferring the data, a handset sends a **special RRC message** to RAN
- The RAN immediately releases the RRC connection and lets the handset go to IDLE
- Fast Dormancy dramatically reduces the tail time, saving radio resources and battery life
- Fast Dormancy has been supported in some devices (e.g., Google Nexus One) in **application-agnostic** manner



# Use Fast Dormancy to Enhance Chunk Mode

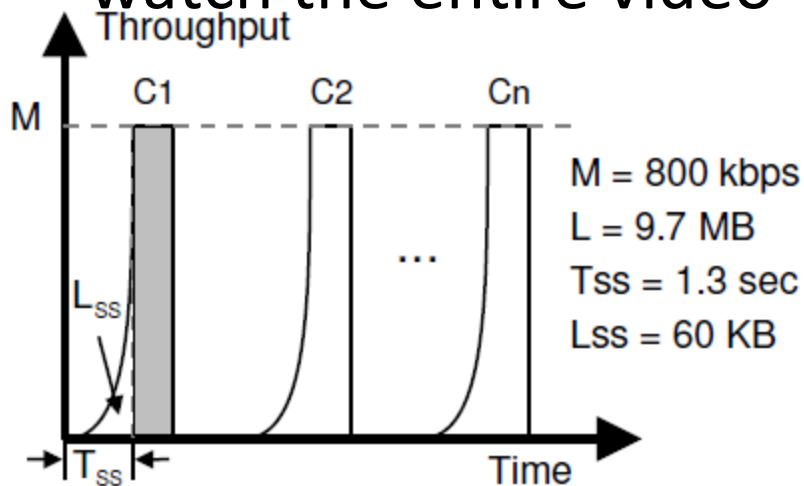
Example: YouTube

- YouTube video streaming
  - Collect a 10-min YouTube trace using Android G2 of Carrier 2.
  - Traffic pattern
    - First 10 sec: maximal bw is utilized
    - Next 30 sec: constant bitrate of 400kbps
    - Remaining: transmit intermittently with the inter-burst time between 3~5 s.
  - **Under-utilization of network bandwidth** causes its long DCH occupation time.
    - Energy/radio resource efficiency is much **worse** than Pandora



# Use Fast Dormancy to Enhance Chunk Mode (Cont'd)

- Proposed traffic pattern: **Chunk Mode**
  - The video content is split into  $n$  chunks  $C_1, \dots, C_n$
  - Each transmitted at the highest bit rate.
  - $n$  should not be too small as users often do not watch the entire video

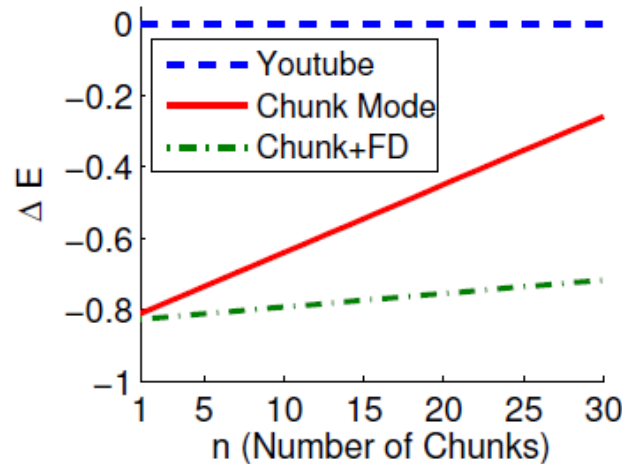
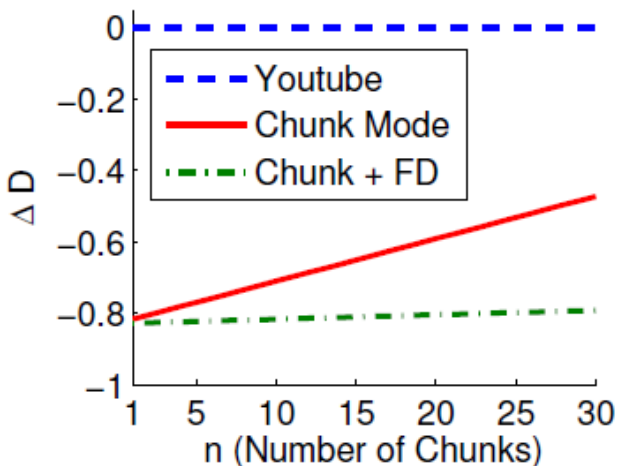


Para	Meaning
M	Maximal BW
L	Content size
$T_{ss}$	Slow start duration
$L_{ss}$	Bytes transferred in slow start

**How to eliminate the Tail for each chunk?  
Using Fast Dormancy**

# Use Fast Dormancy to Enhance Chunk Mode (Cont'd)

- Invoke fast dormancy at the end of each chunk
  - To immediately release radio resources (assuming



ists)

	Relative Change of...
$\Delta E$	Radio Energy
$\Delta D$	DCH Occupation Time

**Chunk Mode: Save 80% of DCH occupation time and radio energy for YouTube**

**Fast Dormancy: Keep  $\Delta D$  and  $\Delta E$  almost constant regardless of # of chunks.**

# Conclusion

- ARO helps developers design **cellular-friendly** smartphone applications by providing **visibility** of radio resource and energy utilization.
- Cellular friendly techniques ([http://developer.att.com/home/develop/referencesandtutorials/networkapibestpractices/Top\\_Radio\\_Resource\\_Issues\\_in\\_Mobile\\_Application\\_Development.pdf](http://developer.att.com/home/develop/referencesandtutorials/networkapibestpractices/Top_Radio_Resource_Issues_in_Mobile_Application_Development.pdf))
  - Group multiple simultaneous connections from the same server
  - Batching and piggybacking
  - Close unnecessary TCP connections early
  - Offloading to WiFi when possible (ms setup rather than 2sec)
  - Caching and avoid duplicate content
  - Prefetching intelligently
  - Access peripherals judiciously
- Try out the ARO tool at:
  - <http://developer.att.com/developer/forward.jsp?passedItemId=9700312>

# Questions?