Aalto University
School of Science

T-110.6120 Energy-efficient Mobile Computing

Energy-efficient wireless networking

23.4.2014

Matti Siekkinen

Outline

- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
- Conclusions

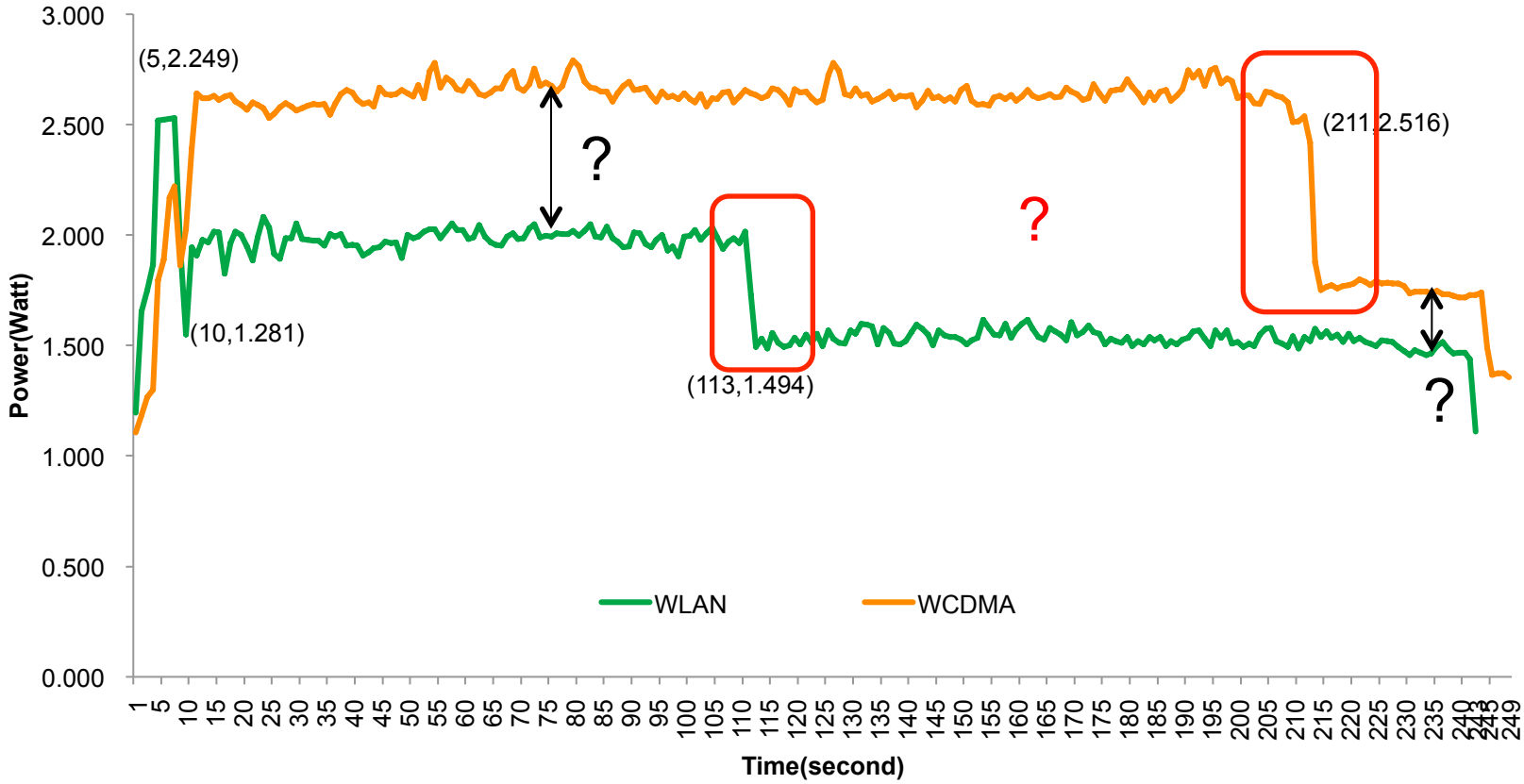
Energy efficiency of wireless and mobile networking

- In short: Resulting battery life when using smartphone to access the Internet
- Two concepts
  - Wireless communication: use radio(s) to communicate
  - Mobile networking: move while communicating
- Using radio requires a certain amount of power
  - How much depends on the type of wireless technology
    - Basically the PHY and MAC layers
- Being mobile means that this power is not constant
  - We'll come to the causes and consequences later on…

Questions, questions…

- Q: Which one is more energy efficient?
  - 3G, WLAN, or LTE?
  - Lumia 920 or iPhone 5?
  - P2P or C/S?
  - ...
- A: It depends...

A glance at power consumption…



Watching YouTube from N95

Energy efficiency of wireless and mobile networking

- Energy efficiency: Spend as few Joules over a period of time as possible
  - Minimal average power consumption
- How to improve the efficiency by means of software?
  - Switch off unnecessary hardware
    - Some of the radio circuitry
  - Increase the number of bits transmitted/received per Joule spent
  - Reduce the number of bits to transmit/receive

Outline

- Intro
- Standard power management techniques
  - Wi-Fi
  - Cellular networks
  - Tail energy
- Beyond standard techniques
- Example optimizations
- Conclusions

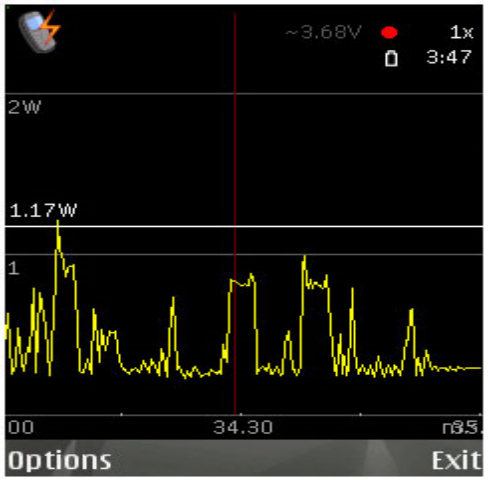
Standard power management techniques

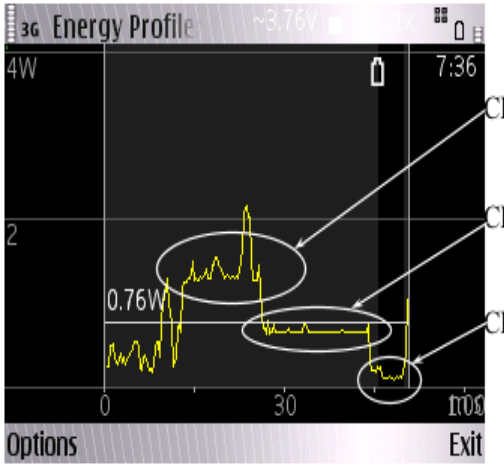
- Operation defined in a standard document
- Usually
  - implemented by each and every device
  - requires cooperation between mobile device and the network
- Wi-Fi: IEEE 802.11 standard
- Cellular networks: 3GPP releases
  - UMTS (3G, Rel-99)
  - LTE (4G, Rel-8)

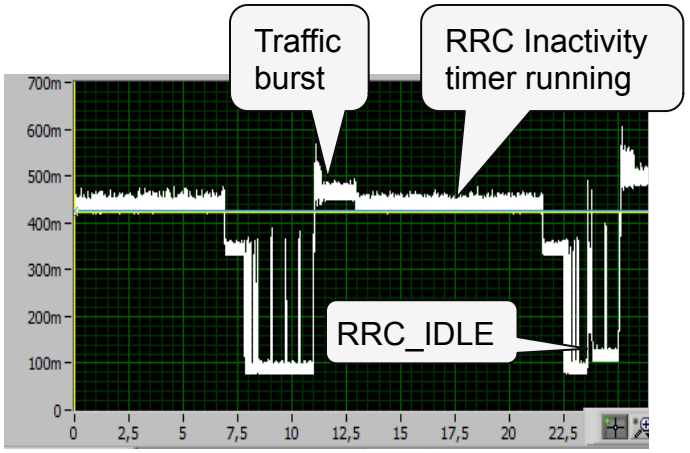
WNI states and transitions

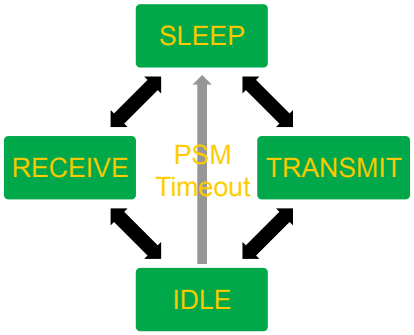
- Management of wireless network interface happens through different states
  - Set of states are technology specific
- WNI transitions from state to another according to some rules
  - Promotions based on traffic demand
  - Demotions usually timer specified
- What states?
  - E.g. receive, idle, and sleep in WiFi
  - Correspond to specific modes of the hardware
  - CELL_DCH, CELL_PCH etc. for 3g
    - Correspond to different kind of resource allocation (i.e. channel type)
- States have different power characteristics
  - Part of circuitry can be powered off at run time (sleep)

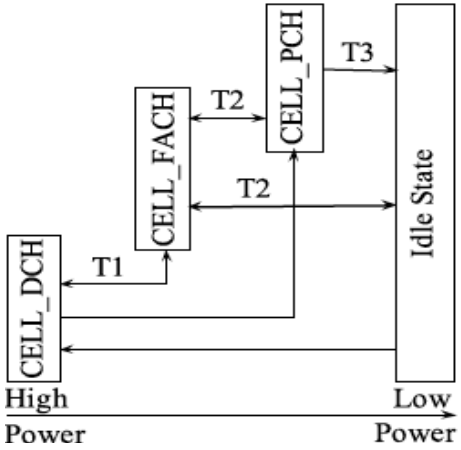
Wi-Fi, 3G, LTE: different power states

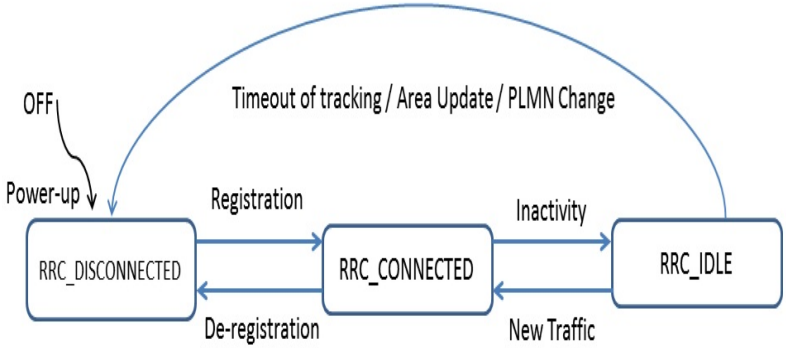












Outline

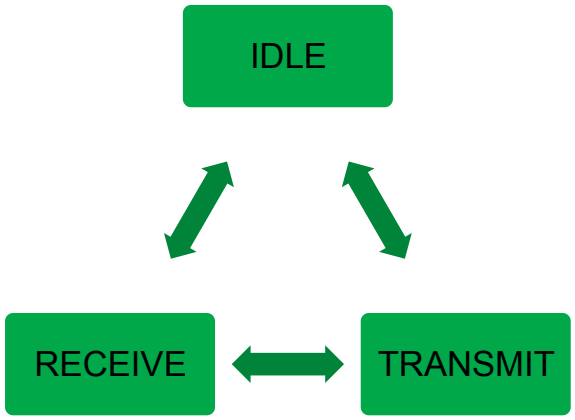
- Intro
- Standard power management techniques
  - Wi-Fi
  - Cellular networks
  - Tail energy
- Beyond standard techniques
- Example optimizations

Wi-Fi (802.11) power consumption

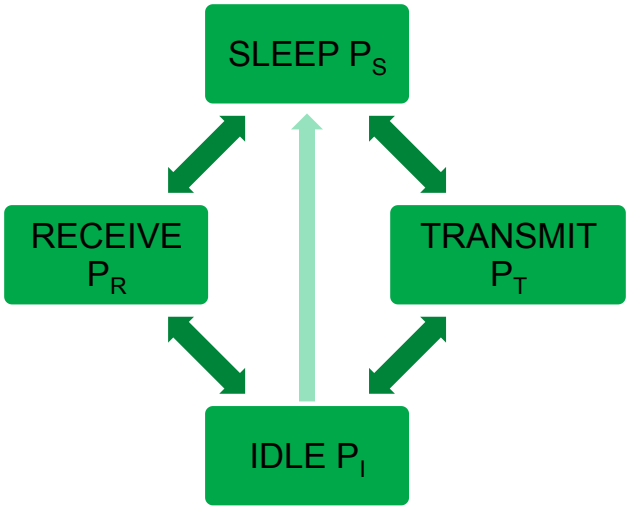
- Power consumption depends on operating mode

  Energy = Power(operating mode) Duration(operating mode)*

Continuously Active Mode (CAM)



Power Saving Mode(PSM)



Wi-Fi operating modes and power

- Order of magnitude less power drawn in sleep state

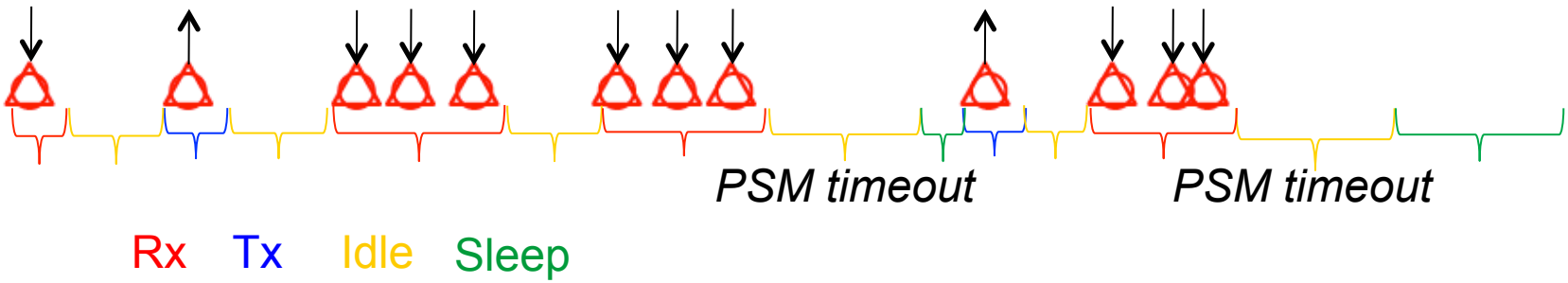
WNI operating mode	Average Power (W)		
	Nokia N810	HTC G1	Nokia N95
IDLE	0.884	0.650	1.038
SLEEP	0.042	0.068	0.088
TRANSMIT	1.258	1.097	1.687
RECEIVE	1.181	0.900	1.585

Wi-Fi power saving

- Allows (part of) Rx/Tx circuitry to be temporarily shut down
- Coordinated with the AP
  1. Node-to-AP: “I am going to sleep until next beacon frame”
  2. AP knows not to transmit frames to this node, buffers them
  3. Node wakes up before next beacon frame
  4. Beacon frame: contains list of mobiles with AP-to-mobile frames waiting to be sent
     - Traffic Indication Map (TIM)
  5. Any frames buffered for the node?
     - Yes → request for them from AP and stay awake until received
     - No → go back to sleep until next beacon frame

Wi-Fi power saving (cont.)

- Standard PSM is poison for interactive applications
  - Frequent transitions to and from sleep mode adds lot of delay
- "Adaptive" version used in practice
  - Use timer: if no frames for 100-200ms, then sleep
  - Timer value is device specific

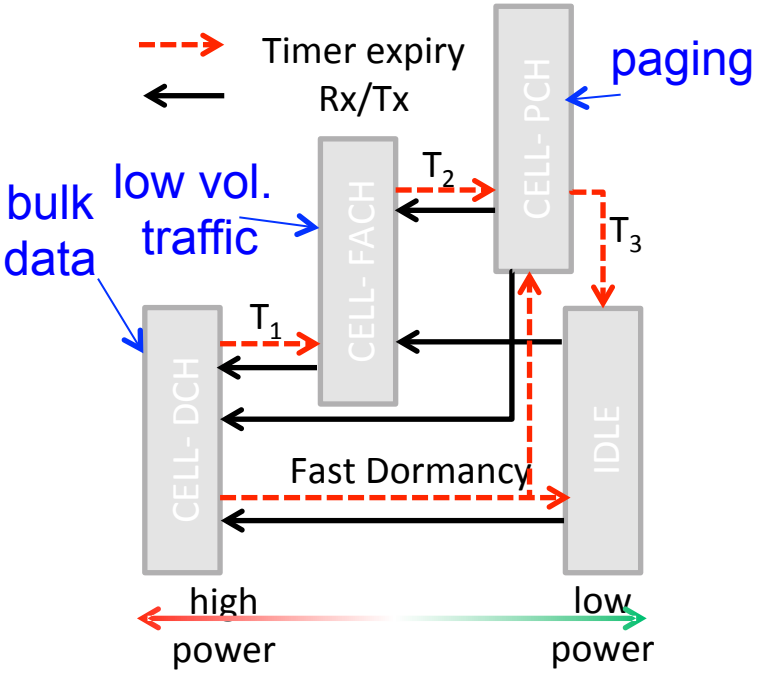


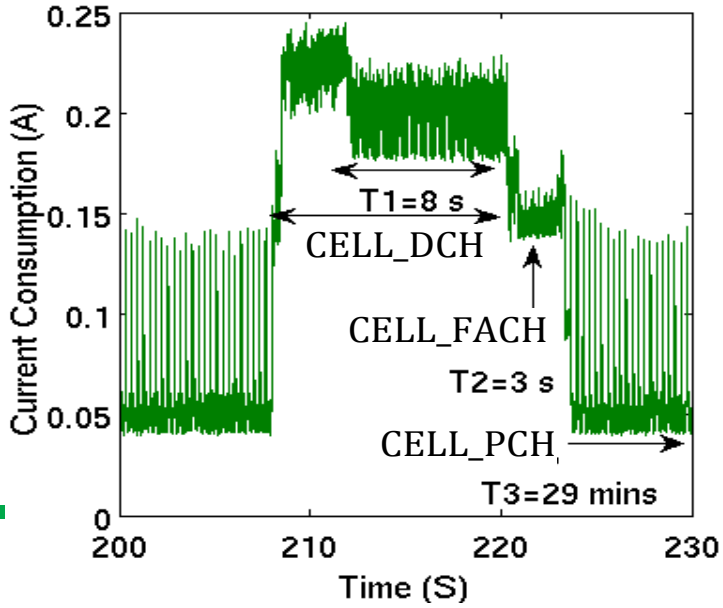
Outline

- Intro
- Standard power management techniques
  - Wi-Fi
  - Cellular networks
  - Tail energy
- Beyond standard techniques
- Example optimizations
- Conclusions

3G power management

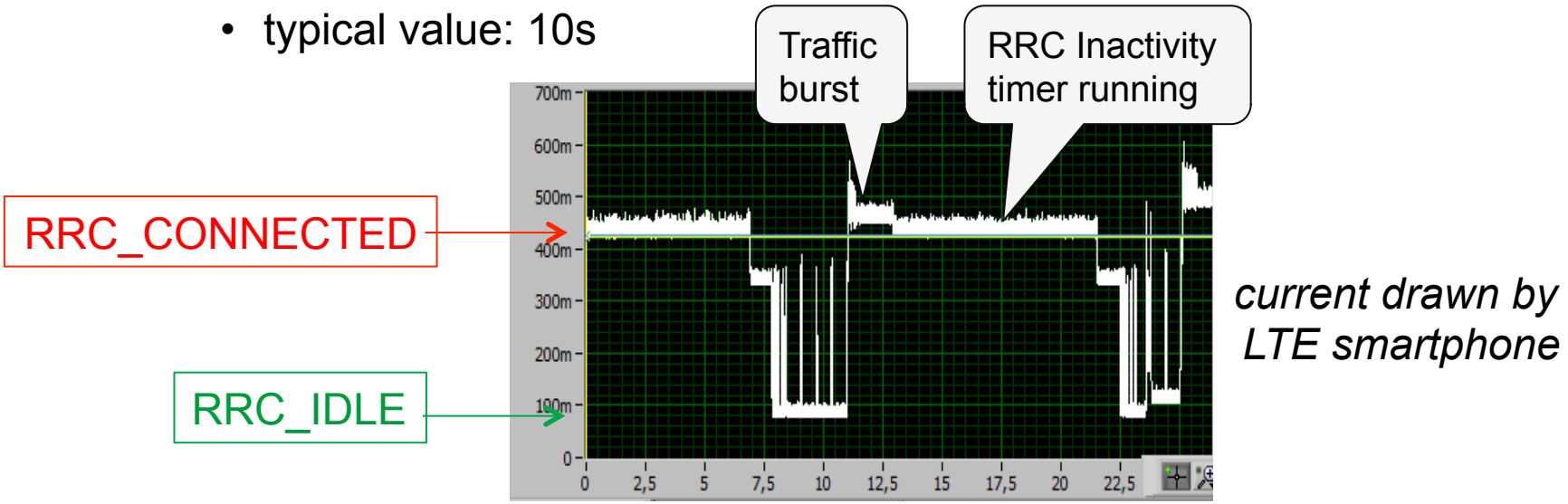
- Radio resources (channel usage) controlled by the Radio Resource Control (RRC) protocol
  - Consequently, power consumption of a mobile phone too
- Four states and three inactivity timers
  - States correspond to transport channels
    - Dedicated channel (DCH), Forward access channel (FACH), Paging channel (PCH)
  - In practice, timers are not just single parameter
    - e.g. track average nb of bytes over time window and use thresholds
    - typically at least a few seconds long
  - Operator controlled, phone cannot change





LTE power management

- Same concept than in 3G
- Simplified RRC protocol
  - Only two states: RRC_CONNECTED and RRC_IDLE
  - Inactivity timer to switch to RRC_IDLE
    - typical value: 10s



current drawn by LTE smartphone

Outline

- Intro
- Standard power management techniques
  - Wi-Fi
  - Cellular networks
  - Tail energy
- Beyond standard techniques
- Example optimizations
- Conclusions

Tail energy

- All wireless network interfaces exhibit *tail energy*
  - Energy spent being idle with radio on → wasted energy
- Due to *inactivity timers*
  - Mandate how long radio remains in active state (rx on) before state transition to inactive state (rx (partly) off)
- Timers are necessary
  - Sporadic communication patterns might lead to very frequent transitions
  - State changes require signaling between phone and base station
    - Transmitted on shared channel with limited capacity
    - Signaling traffic volumes must be limited
  - Also switching between hardware modes adds some delay
- Timer values vary between technology
  - Wifi≈100-200ms
  - 3G and LTE: in the order of seconds (varies between ISPs)

How to minimize tail energy?

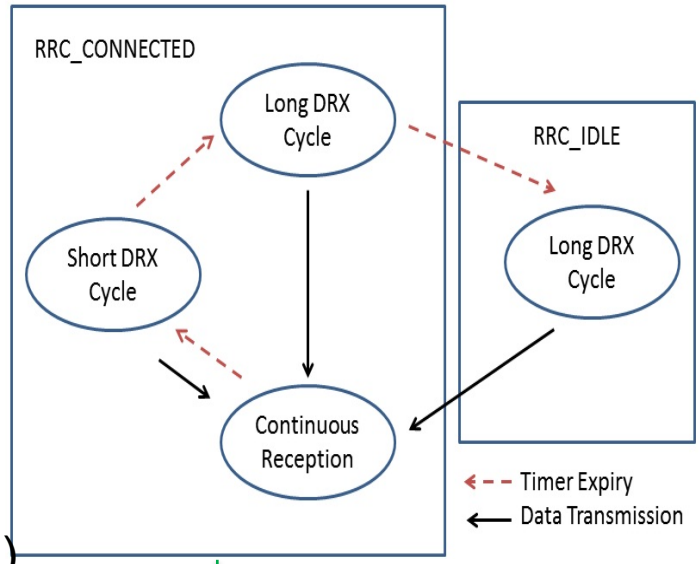
- Wi-Fi tail is already short
  - No need for specific mechanisms
- Fast Dormancy for 3G
  - Cuts tail duration down to 3-5s
- DRX/DTX for LTE
  - Especially cDRX/cDTX: connected mode discontinuous reception/ transmission
  - Typically cuts tail duration down to a few hundred milliseconds
- 3G has also CPC
  - Continuous packet connectivity
  - Similar to LTE's cDRX/DTX
  - Introduced in Rel-7 but often not (yet?) fully supported by deployed networks and devices

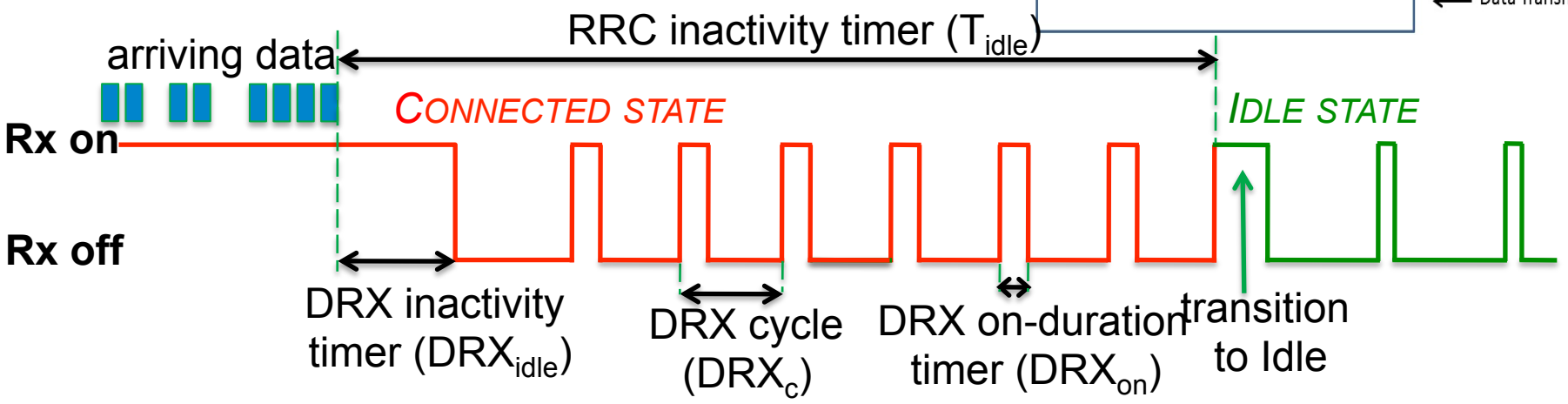
Fast Dormancy (3G)

- Two flavors: legacy and standard
- Legacy came first and is phasing out
  - Phone transmits SIGNALING CONNECTION RELEASE INDICATION msg → tears down PS signaling connection
    - Normally phone uses to communicate some error conditions
  - Good: Results in immediate transition to low power IDLE
  - Bad: new communication requires re-establishing of signaling connection → frequent use causes signaling storms
- Standard FD in Rel-8 is network controlled
  - Phone requests network to transition it into an appropriate state (e.g. CELL_PCH)
  - Network either allows or denies and decides appropriate state
    - E.g. too frequent requests are rejected

Connected mode DRX/DTX (LTE)

- DRX works in LTE's connected state
  - Hence, also called cDRX (connected mode DRX)
- DRX operates in cycles
  - Check periodically if new data is waiting
  - Very similar to PSM in 802.11





Outline

- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
- Conclusions

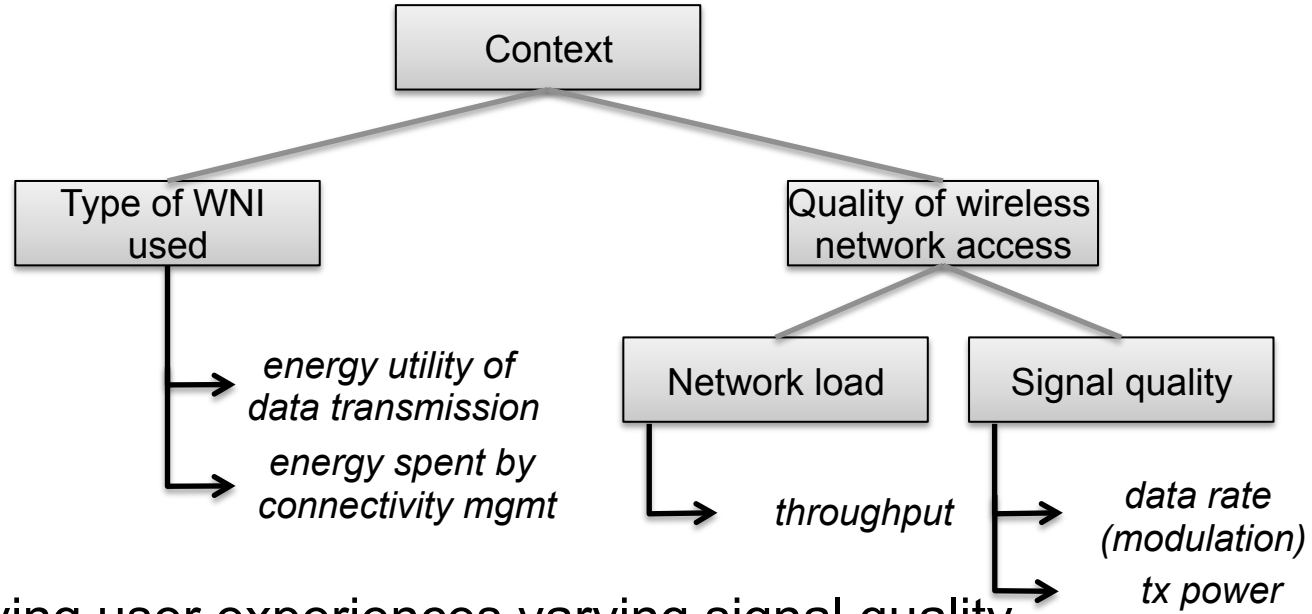
Is there room for optimization?

- Typical application consumes a lot more energy than is strictly necessary
  - Even with standard power saving mechanisms
- Three reasons:
  1. Radio hardware is not perfectly power proportional to the offered load
  2. Energy utility of wireless communication is context dependent
  3. The underlying hardware power management mechanisms are rarely optimal for the applications being used

Power (dis)proportionality

- Power draw does not scale linearly with amount of work done
  - Bits transmitted/received per Joules spent typically increases with data rate
- Idle power consumed by hardware just being powered on
  - That constant power added regardless of transmission rate
- Idle power takes a larger share of slower transmission
  - Over-the-air (OTA) data rate of wireless channel != throughput
  - Each packet may be transmitted continuously at OTA rate but idle time in between packets spent as tail energy

Energy efficiency is context dependent



- Moving user experiences varying signal quality
  - Dynamic switching of modulation based on SNR → data rate changes
  - Poor SNR → link layer retransmissions
  - Worse SNR requires more transmit power
    - Affects also reception (cont. tx of signaling msgs)
- Static user may experience varying network load
  - Other users take up air time

Limits of underlying hardware power management mechanisms

- Hardware power management mechanisms implemented at PHY/MAC layers
  - The protocol stack layers that interface the hardware
- Separation of concerns in the layered protocol design
  - Layer is concerned only about its own responsibilities and functionalities
  - Cross-layer protocols are not so common
    - At least not those that cross the stack up to application layer
- → power management mechanisms are completely unaware of application behavior
  - Same mechanism regardless of the type of application

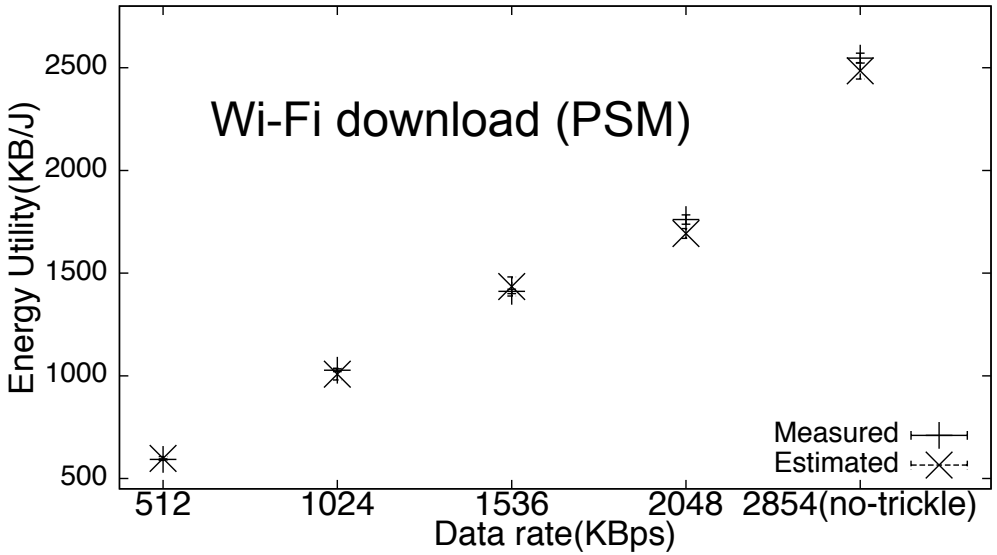
How can we do better?

- Traffic scheduling
  - Shape traffic to improve energy utility (bits per joules)
  - Reduce energy spent due to contention
  - Take context dependency into account
  - Handling background traffic energy efficiently
- Smart use of wireless network interfaces
  - Smartphones have many
  - Use always the most energy efficient one
  - Often requires ability to predict connectivity
- Application specific optimizations

Outline

- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
  - Traffic shaping in multimedia streaming
  - Optimized prefetching of video streaming
  - Context-aware scheduling of transfers
- Conclusions

Energy utility



Measurements using Samsung Nexus S

- Energy utility improves with data rate
  → should always transmit as fast as possible

Energy consumed by mobile streaming

- Mobile media streaming drains battery quickly
  - Constant bit rate multimedia traffic is not energy friendly
  - Forces the network interface to be active all the time

Mobile Internet Radio power draw on E-71 (TCP-based streaming)

Datarate (kBps)	Start-up Time (s)	WLAN power (W)		3G power (W)	
		PSM	CAM	48kBps	2Mbps
8	18	0.53	1.06	1.30	1.30
16	10	0.99	1.07	1.30	1.30
24	10	1.04	1.07	1.27	1.35

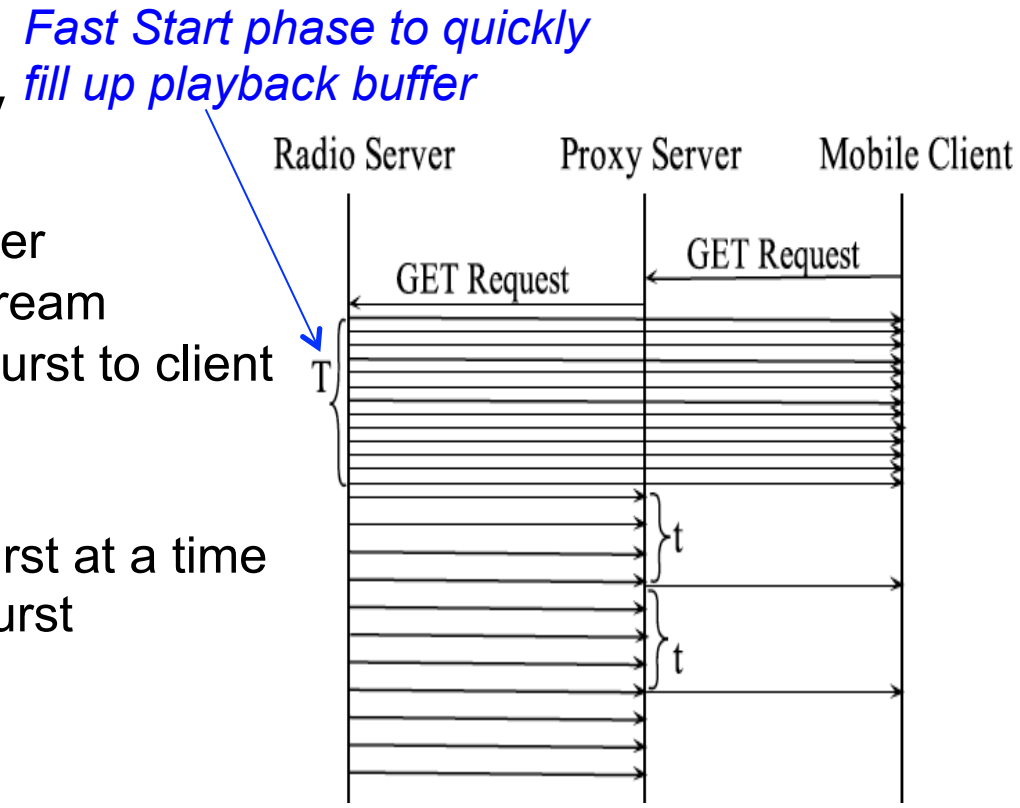
- Idea: Shape traffic into bursts so that it is more energy efficient to receive
  - Bursts sent at high data rate
  - “Race to sleep”

Multimedia Traffic Shaping with Proxy

- Client sends request to proxy
- Proxy
  - forwards request to radio server
  - receives and buffers media stream
  - repeatedly sends in a single burst to client
- 802.11
  - PSM is enabled
  - WNI wakes up to receive a burst at a time
  - Waste only one timeout per burst
- 3G & LTE
  - Long enough burst interval

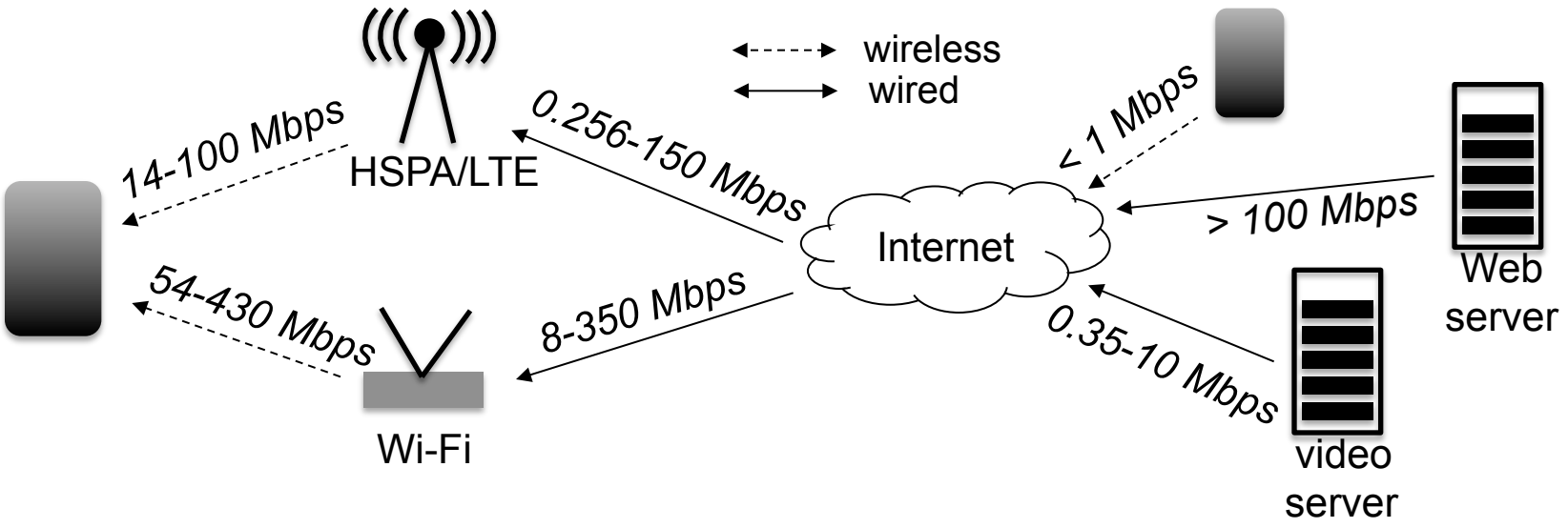
  → inactivity timers expire

  → switch to lower power state or activate DRX in between bursts



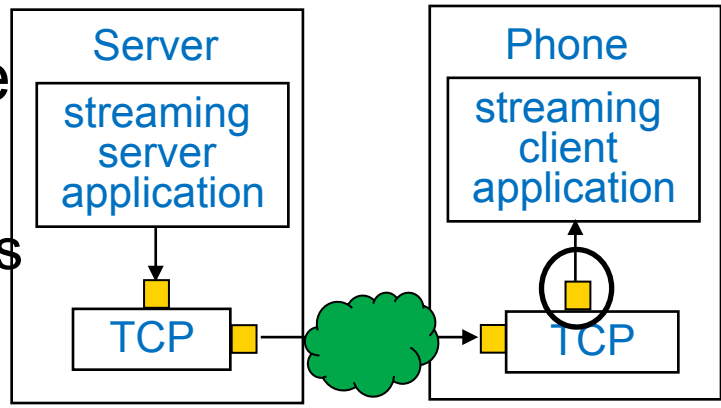
Where is the free lunch?

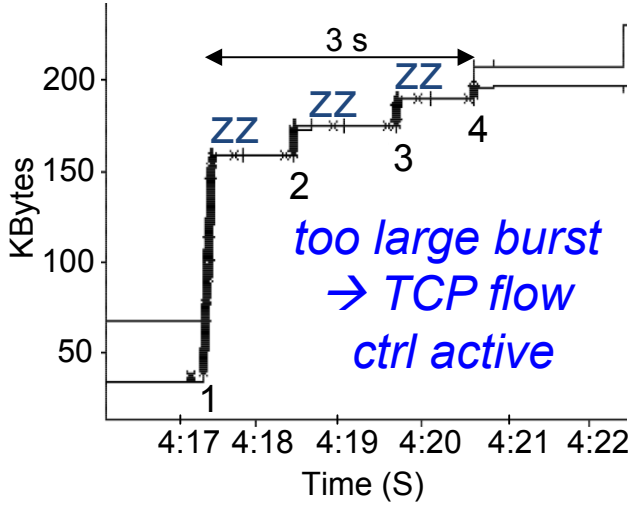
- Typical multimedia traffic leaves lots of network capacity unused
- Several vantage points possible for traffic shaper proxy placement



What is the right burst size?

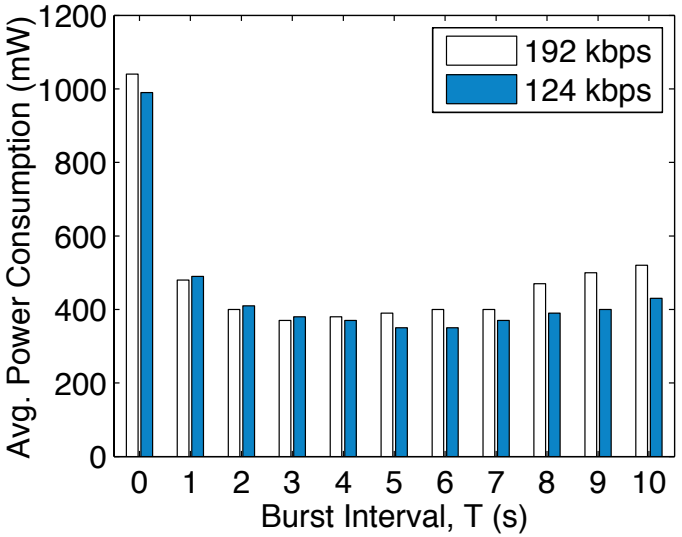
- Intuition: Use as large as burst size as possible
  - Maximize sleep time in between bursts
- Problem: TCP receive buffer is of limited size
  - Too large burst will not fit entirely → TCP flow control kicks in
  - Buffer is drained at stream encoding rate → Excess content will be received at that rate
    - Lower energy utility





What is the right burst size?

- Burst size that offer maximal energy savings exists
  - Option 1: Make burst size match exactly receive buffer size
    - Max burst size = playback buffer size +TCP receive buffer size
  - Option 2: Max burst interval & size limited by amount of initially buffered content
    - Cannot let the playback buffer run dry
- Optimality of such burst size is easy to prove
  - Check: *M. Hoque et al. Saving Energy in Mobile Devices for On-Demand Multimedia Streaming - A cross-layer Approach. ACM TOMCCAP. 2014.*

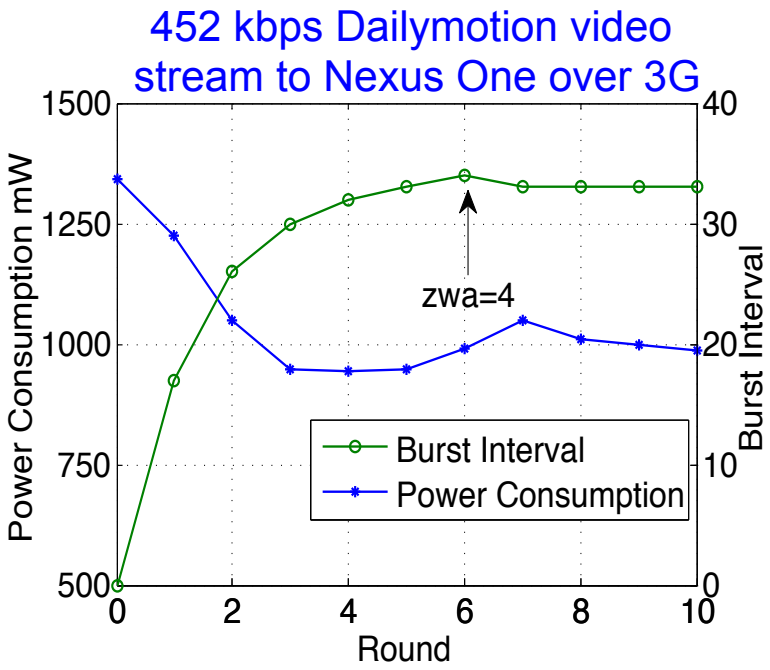


How to find the optimal burst size?

- Proxy does not know client's TCP receive buffer size
- Could design a protocol where client informs proxy server
  - Need custom streaming client software → bad idea
- Insight: TCP flow control messages indicate too large burst size
  - Proxy receives TCP zero window advertisements from client
  - Provides transparent way of identifying too large burst size
- How to probe for right size?
  - Linearly increasing burst size can take a long time
  - Use binary search instead

How to find the optimal burst size?

- Initially
  - Set max burst size according to initially buffered data
    - Proxy calculates it during the Fast Start period
  - Start from min=0
- For each burst
  - Set new burst size to *(max-min)/2*
  - Check whether received any ZWAs
    - No → set min=current burst size
    - Yes → set max=current burst size and revert back to previous burst size
- Stop search when max-min equals minimum increment
  - E.g. 1s worth of content



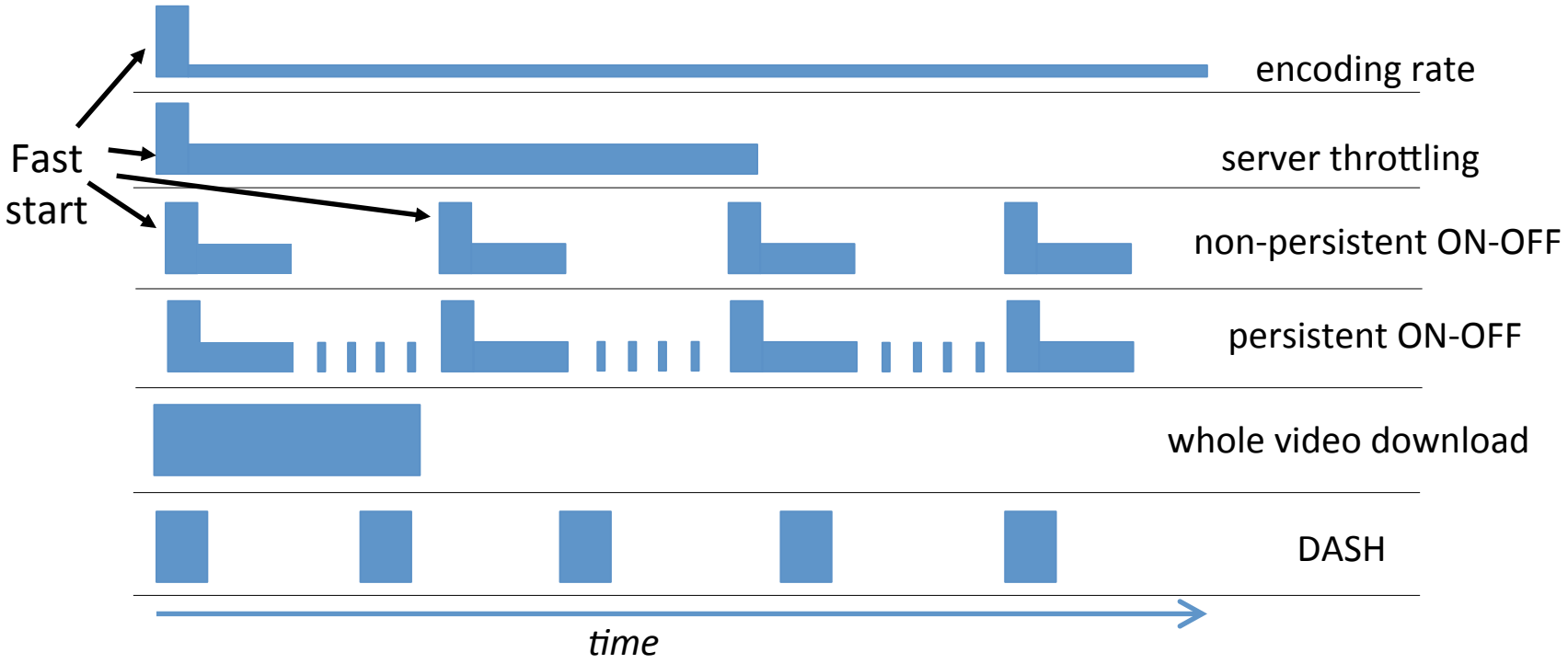
How much energy can be saved?

- Overall large savings possible
  - audio: 36%-65%, video: 20%-55%
- Savings depend largely on network type and parameters
  - 3G/LTE have longer inactivity timers than Wi-Fi
  - Parameters (timers and DRX) determine the tail energy that can be saved
- Stream rate matters as well
  - Bursting lower rate stream yields larger savings
- Smaller savings with video streaming compared to audio
  - Display draws significant amount of power
  - Video decoding is more work than audio decoding

Outline

- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
  - Traffic shaping in multimedia streaming
  - Optimized prefetching of video streaming
  - Context-aware scheduling of transfers
- Conclusions

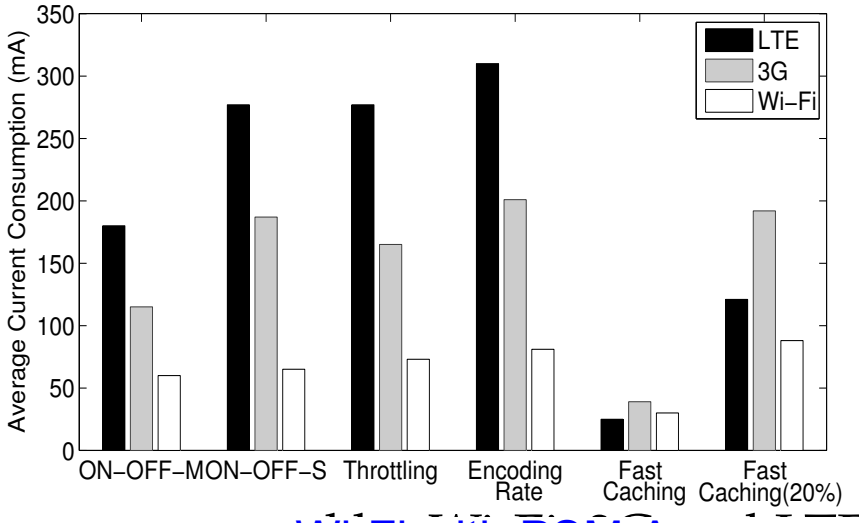
On-demand mobile video stream delivery



- Different strategies to deliver video stream content to client
  - Caused by client software+streaming service combinations
  - Leads to very different energy consumption

Energy consumption of different strategies

- Several sources of energy inefficiency
  - Underutilization of the capacity
  - Auxiliary TCP control traffic (ON-OFF-S)
  - Tail energy in non-continuous content reception
- Fast Caching seems best
  - But users typically abandon viewing!
  - No longer best when aborting after 20%
  - Too aggressive prefetching → download content that will never be watched



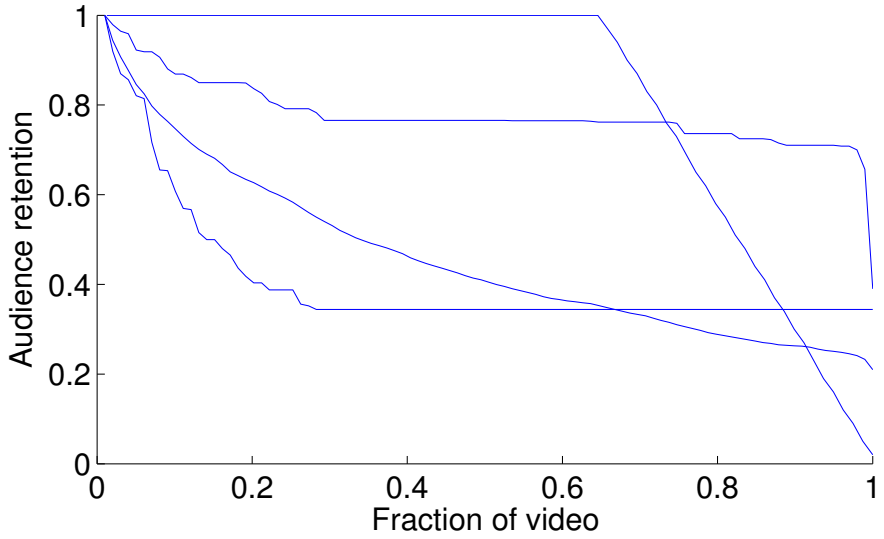
Wi-Fi with PSM-A
3G (HSPA) with FD
LTE without DRX

Optimizing content delivery strategy

- Must strike a balance between two sources of energy waste:
  - Prefetch content in large chunks in order to minimize the tail energy
  - Limit chunk size in order to reduce the amount of downloaded content that will never be viewed
- Need an estimate of when a user will abandon viewing the video

  → use viewing statistics

Video viewer retention



- Video clip specific statistics give insights about how a new user would view the video
  - E.g.: poor content → likely to abandon early
- YouTube collects such statistics
  - Available only to “video owner”

eSchedule algorithm

- Algorithm that calculates optimal download schedule
- Input:
  - Viewer retention data for the video clip
    - R_i: “Which fraction of users viewed until time step i”
  - Power consumption characteristics of WNI used (P_{rx}, E_{tail})
  - Video stream rate (r_{enc}) and bulk transfer capacity (r_{dl})
- Output: dl schedule S that minimizes expected value of energy waste
  - Exp amount of energy spent by downloading content that won’t be viewed
  - Estimate of total tail energy expenditure
  - S is a concatenation of chunk sizes (n-tuple): $S=(T_1, T_2, \ldots, T_n)$

eSchedule algorithm: problem formulation

Probability that user interrupts during a discrete time step *j*:

$$\begin{aligned} p_{abd}(j) &= p(\text{“abandon at j”}) = R_{j-1}p_j(X=0) \\ &= R_{j-1}(1-p_j(X=1)) = R_{j-1} - R_j \end{aligned}$$

computed from viewer retention data (R_i)

Exp. value of unnecessarily downloaded content for a given chunk (starts at i, dur T):

$$\mathbb{E}[B_{waste}(i,T)] = \sum_{k=1}^{\lceil \frac{T \times r_{enc}}{r_{dl}} \rceil} p_{abd}(i+k)\left[k \times r_{dl} - k \times r_{enc}\right] + \sum_{k=\lceil \frac{T \times r_{enc}}{r_{dl}} \rceil + 1}^{T} p_{abd}(i+k)\left[T \times r_{enc} - k \times r_{enc}\right]$$

amount of content in buffer at given time instance

Exp. value of energy waste for a single chunk:

$$\mathbb{E}[E_{waste}(i,T)] = E_{tail}(T) + \frac{\mathbb{E}[B_{waste}(i,T)]}{r_{dl}} \times P_{rx}(r_{dl})$$

tail energy + unnecessary dl energy

Exp. value of energy waste for entire dl schedule:

$$\begin{aligned} \mathbb{E}[E_{waste}(1,S)] &= E_{tail}(T_1) + \\ &\sum_{j=2}^{n}\left[(1 - \sum_{k=1}^{\sum_{l=1}^{j-1} T_l} p_{abd}(k)) \times E_{tail}(T_j)\right] + \\ &\sum_{j=1}^{n}\left[\mathbb{E}\left[B_{waste}(\sum_{k=1}^{j-1} T_k, T_j)\right]\right] \times \frac{P_{rx}(r_{dl})}{r_{dl}} \end{aligned}$$

discount future tail energy that may not happen

eSchedule algorithm: solving optimum

- How to find schedule S that minimizes $\mathbb{E}[E_{waste}(1, S)]$?
- Brute force search does not scale
  - Number of possible solutions grows exponentially with video length (1 min video → over 2 M possibilities)
- Use dynamic programming
  - Iterative algorithm
  - Compute and store solutions to subproblems
    - Optimal schedule for a part of the video
  - Use stored solutions in subsequent iterations
    - Avoid redundant calculations
  - Feasible complexity: $O(n^2)$ where n is video length in min size chunks

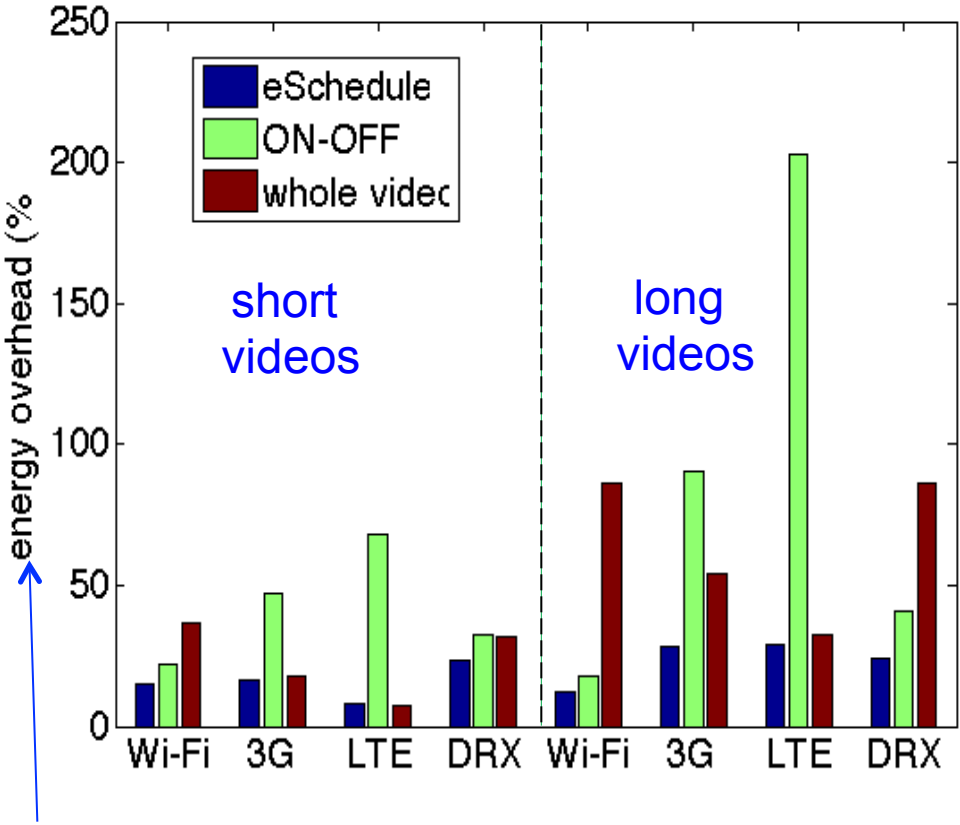
B. Jackson et al., "An algorithm for optimal partitioning of data on an interval," Signal Processing Letters, IEEE, vol. 12, no. 2, pp. 105–108, Feb 2005.

Performance evaluation

- Simulations using Matlab
- Short and long videos
  - 5 min threshold
  - 16 videos in total
- Retention and other data from real YouTube videos
- Abandoning time randomly drawn from the probability distribution given by retention
- Power models based on measurements with Samsung Galaxy S3 LTE
- Video download schedule computed using eSchedule and total energy consumed was calculated for each session
  - Compare to ON-OFF and whole video download

Performance evaluation

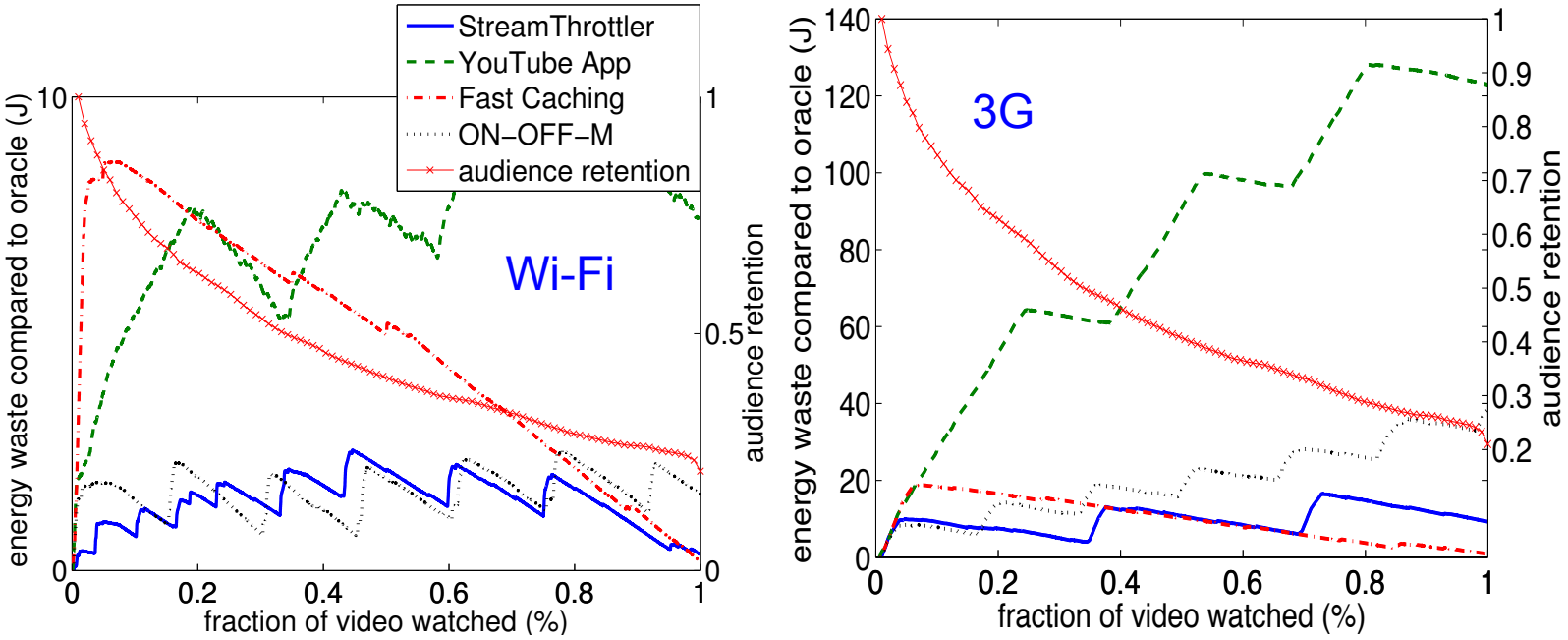
- ON-OFF strategy is fairly good when tail energy is small
  - Wi-Fi and LTE with DRX
- Whole video dl pretty good with short videos and large tail energy
- eSchedule roughly halves the energy overhead



compared to oracle (always downloads the right amount in one shot)

Android app

- Implemented also a prototype app for Android



Outline

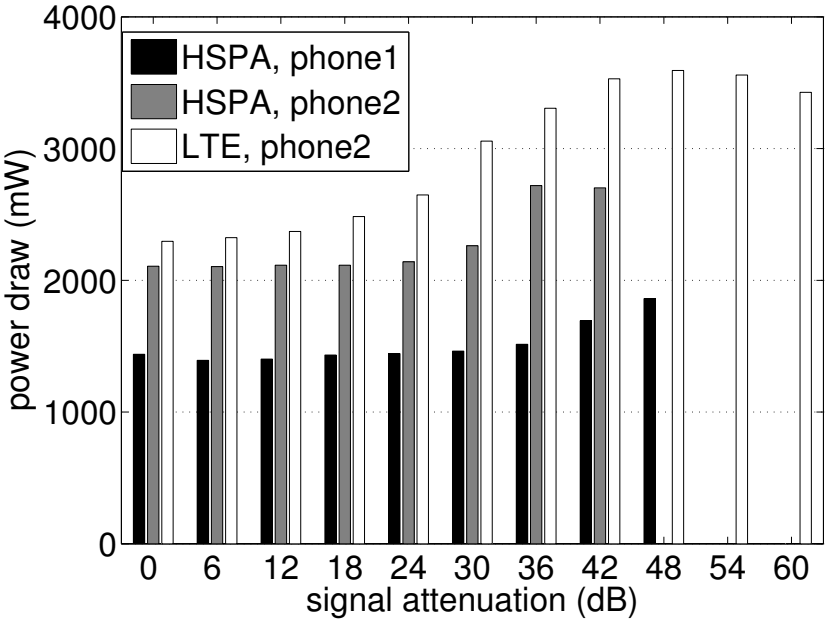
- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
  - Traffic shaping in multimedia streaming
  - Optimized prefetching of video streaming
  - Context-aware scheduling of transfers
- Conclusions

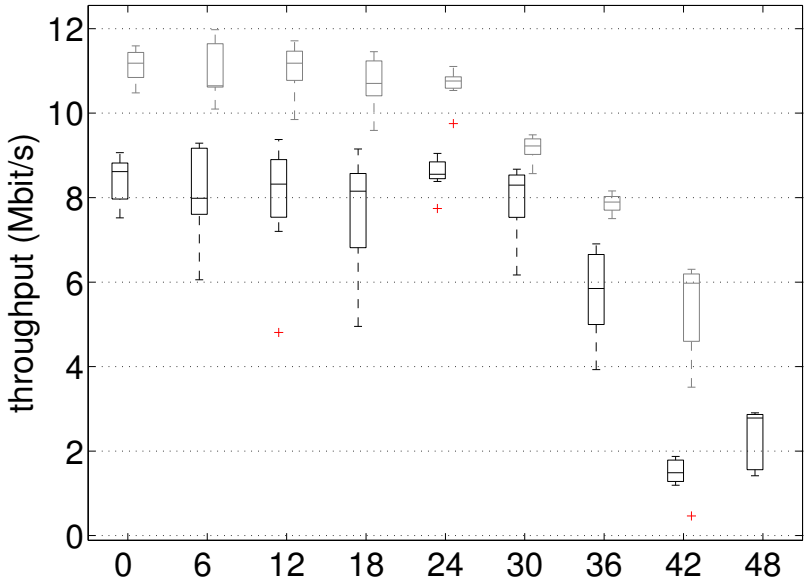
Context-aware scheduling

- We'll look at Bartendr as an example
  - Aaron Schulman et al. *Bartendr: a practical approach to energy-aware cellular data scheduling.* In ACM MobiCom 2010.
  - Learning to schedule transfers in suitable times

How signal strength impacts energy?

- Two-way impact: throughput and transmit power





Bartendr: basic idea

- Moving phone experiences varying signal strength
- Idea: Schedule transfers to happen at times of good signal strength
  - Save energy by holding those transfers that are not time critical
- Example applications
  - Background sync: 5 min interval sync could be more efficient if done sometime between 4 to 6 min
  - Streaming media: Consume buffer when the signal is weak, prefetch when the signal is strong
- Challenge: How to know when to transmit and when to hold?
  - Need prediction

Bartendr: signal tracks

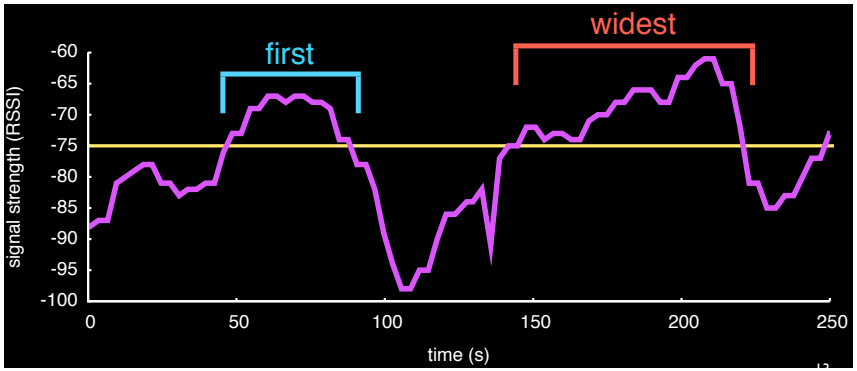
- People tend to move along same paths in their day-to-day life
- Bartendr predicts signal strength for a phone moving along a path
  - Use previous signal measurements captured while traveling along the same path
  - Signal strength measurements are basically energy free
    - phone needs to do it anyway for handoffs
- Assumptions:
  - Phones can store several such *signal tracks* (frequently traveled paths)
  - Phone can infer currently traveled track using mobility prediction

Bartendr: signal strength prediction

- Step 1: find the current position of the phone on the current track
  - GPS draws too much power
  - Find position in track with closest measurement to current one
  - May have many similar strength positions → use also neighbor base station list
- Step 2: predict signal in the future starting from current position
  - Look ahead in previous measurements
  - Speed may differ → continuous update of current position on track

Bartendr: scheduling bg sync

- Schedule next sync based on predicted signal strength
- Sleep in between schedule and sync events → may make errors
- Two threshold-based schemes:
  - *First* above threshold
    - Comes soon → error small
  - *Widest* above threshold
    - Comes later but has larger margin for error



- Simulations suggest 10% energy savings for email sync
  - Widest outperforms first

Bartendr: scheduling stream transfers

- Differs from syncing because continuously awake → can compensate for speed variations in real time
- Results into rather similar optimization problem than we looked at with eSchedule
  - Chunked downloading
  - Energy spending is function of tail energy and energy spent for dl
    - dl energy varies with signal strength
  - Can be solved using dynamic programming too
- Simulations suggest 60% energy savings

LoadSense

- LoadSense is kind of follow-up work to Bartendr
  - Abhijnan Chakraborty et al. *Coordinating cellular background transfers using loadsense.* In ACM MobiCom 2013.
- Solution for scheduling background transfers
- Takes explicitly into account load in the cell
  - Link quality alone is insufficient
  - Load can be passively inferred through *power ratio*
    - Sense total power in the cellular channel and compare it with the power of the pilot signal transmitted by base station
- Schedule bg transfers when sensed load is small
  - Improve energy efficiency by transmitting with higher throughput

Outline

- Intro
- Standard power management techniques
- Beyond standard techniques
- Example optimizations
- Conclusions

What else can be done?

- Smarter (cooperative) scheduling to reduce contention
  - Random access channels (Wi-Fi) can cause energy waste by idle waiting
  - Multiple clients of same AP or multiple APs in range (same ch)
- Leverage alternative low-power radios
  - E.g. Zigbee or Bluetooth in conjunction with Wi-Fi
  - Idea is to always use lowest power radio for the job
  - Discovery of Wi-Fi access points using Bluetooth contact patterns
    - Saves energy spent by Wi-Fi scanning
  - Data transmission with highest possible energy utility
- Synchronize background transfers by different apps
  - Amortize tail energy by batching bg transfers

Summary

- Energy efficiency of wireless networking
  - Depends on wireless access network technology used
  - Depends on context → signal quality and network load
- Standard power management techniques
  - Wi-Fi has PSM
  - Cellular network technologies have RRC
  - Tail energy can be mitigated in cellular nws by using
    - Fast Dormancy (3G)
    - Discontinuous reception (LTE and 3G)
- Optimizing energy efficiency further
  - Traffic scheduling
    - Traffic shaping
    - Context-aware scheduling
  - Smart use of wireless network interfaces
    - Use always the most energy efficient one
  - Application specific optimizations
    - Mitigate mismatch between power mgmt and application behavior