TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012

Tips and Tricks: Designing low power Native and WebApps

Harita Chilukuri and Abhishek Dhanotia

Acknowledgements

- **William Baughman** for his help with the browser analysis
- **Ross Burton & Thomas Wood** for information on Tizen Architecture
- **Tom Baker, Luis “Fernando” Recalde, Raji Shunmuganathan and Yamini Nimmagadda** for reviews and comments

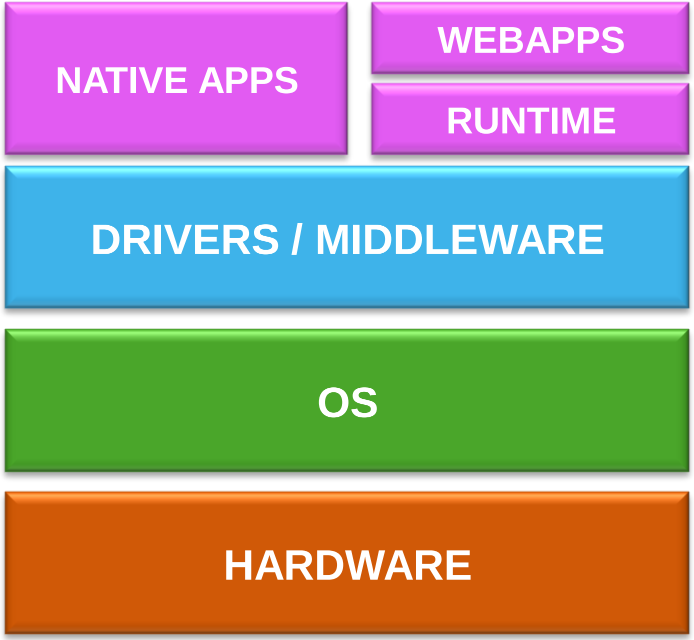
Power – Onus lies on Software too!

Use system resources to provide best User Experience with minimum power

Interfaces with HW components, Independent device power management

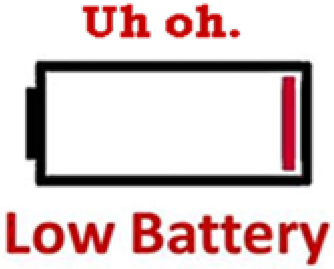
Frequency Governors, CPU Power Management ACPI/RTPM

Provides features for low power Clock Gating, Power Gating, Sleep States



Power – Onus lies on Software too!

- A single bad application can lead to exceeding power budget
- Hardware and OS provide many features for low power
  - Apps need to use them smartly to improve power efficiency
- Good understanding of underlying system can help in designing better apps



Images are properties of their respective owners

Agenda

Power Tools & Metrics

Tips for Low Power Applications

General guidelines

Q&A

Estimating Power - Metrics

- CPU utilization
- Memory bandwidth
- CPU C and P state residencies
- Device D states - For non-CPU components
- S states – system sleep states
- Wakeups, interrupts

Soft metrics can help tune the application for optimal power

Estimating Power - Tools

- CPU utilization
  - Vmstat, Top
  - VTune, Perf for CPU cycles
- Memory bandwidth
  - Vtune
- CPU C and P states, Device D states
  - VTune, Powertop
- Wakeups, Interrupts, Timers
  - Powertop, /proc stats
- Tracing tool in Chrome browser

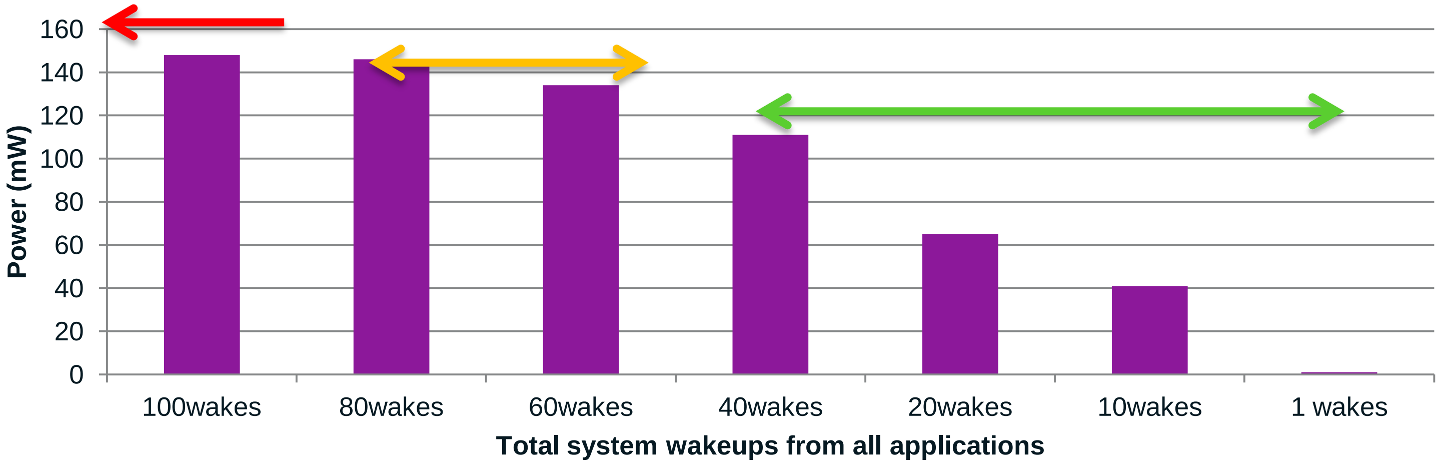
** VTune is an Intel product and can be purchased, others are publicly available Linux tools

* Other names and brands may be claimed as the property of others.

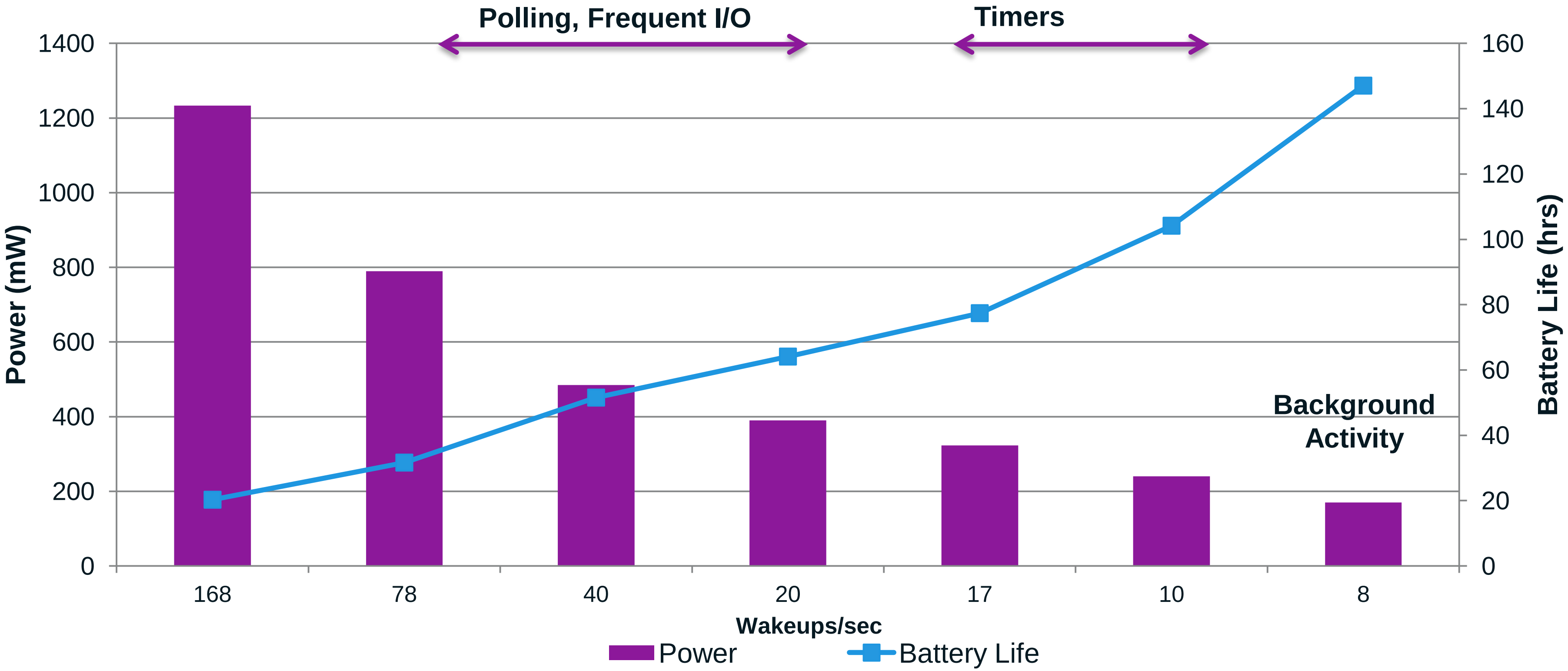
Tip 1:
Minimize wakeups, they are expensive

Tip 1: Minimize wakeups

- Classic case of a single application exhausting the power budget
- Wakeups from each app add up
  - Even if its just 0.5 wakeups/sec



Tip 1: Minimize wakeups



Tip 1: Minimize wakeups

- How to reduce wakeups
  - Polling
  - High frequency timers
  - Avoid frequent I/O

  + Maximize the work done when the system wakes up, batch operations
  + Longer sleep time is better than frequent shorter sleep times

Apps accessing data from peripheral components/sensors

- GPS, Games using accelerometer etc.

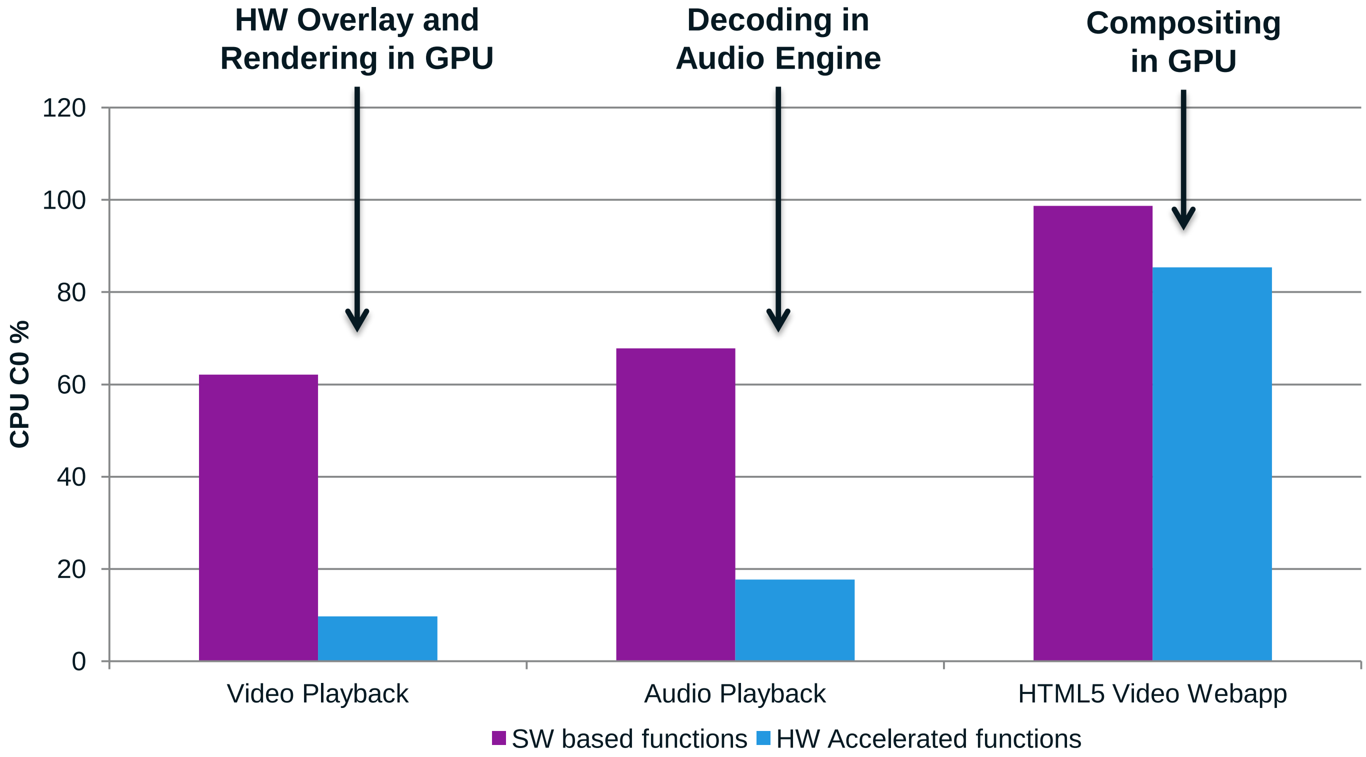
Apps doing periodic updates

- Push notifications instead of polling

Images are properties of their respective owners

Tip 2
Use Hardware Acceleration

Tip 2: Use hardware acceleration



HW Overlay and Rendering in GPU

Decoding in Audio Engine

Compositing in GPU

Tip 2: Use hardware acceleration

Caveats

- Benefits from acceleration vary based on use case
- HW Acceleration improves performance only when

  TIME (memcpy + work in HW) < TIME (work in SW)
- Power

	Software	Hardware Acc
CPU	↑	↓
GPU	↓	↑
Memory	↓	↑

Tip 2: Use hardware acceleration

- Common formats that are usually accelerated or optimized
  - Developers should use these when applicable

Video
MPEG2
MPEG4
H264
VC1

Audio
MP3
AAC

Browser
GPU Accelerated Compositing
GPU Rendering
EGL

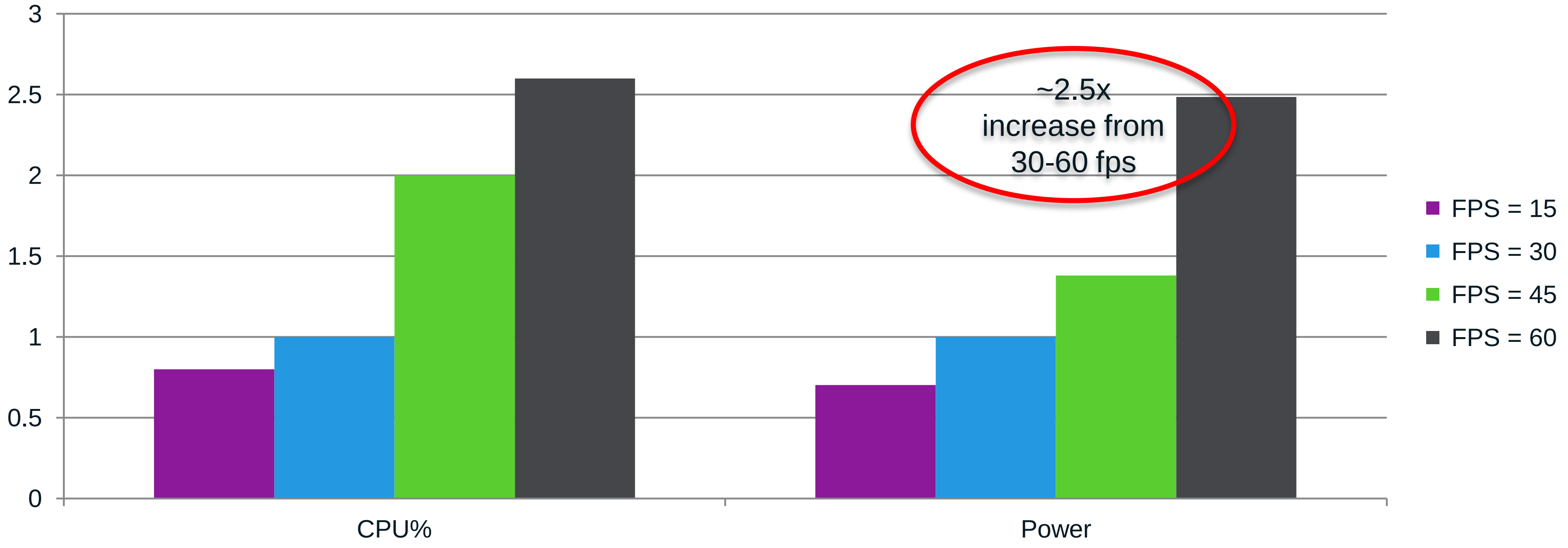
- Media apps, local playback and streaming
- Games

Images are properties of their respective owners

Tip 3:
Don’t shoot for performance beyond what a user can perceive

Tip 3: Performance Vs. Power Tradeoff

- Video playback power use with different frame rates



*** Power here is the difference between system idle and video playback, data normalized to 30 fps.**

Tip3: Performance Vs. Power Tradeoff

– Trying to maximize performance without knowledge of user expectation

+ Make a tradeoff

Media Apps

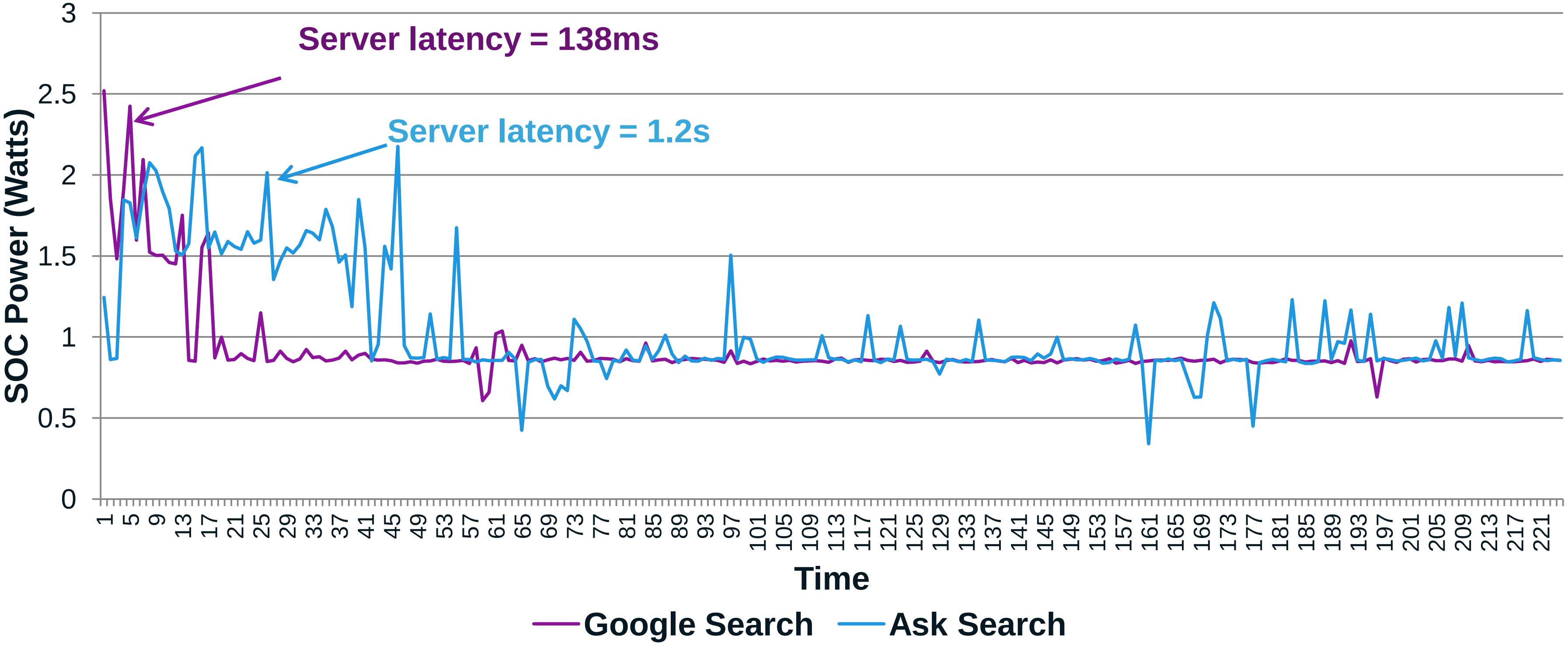
- Videos beyond 30 fps are rarely perceivable on mobile devices
- Audio bit rates beyond audio HW output capacity won’t sound better

Games

- Screen refresh rates are 60/120Hz, frames are dropped beyond that
- 60 fps may be overkill for some applications

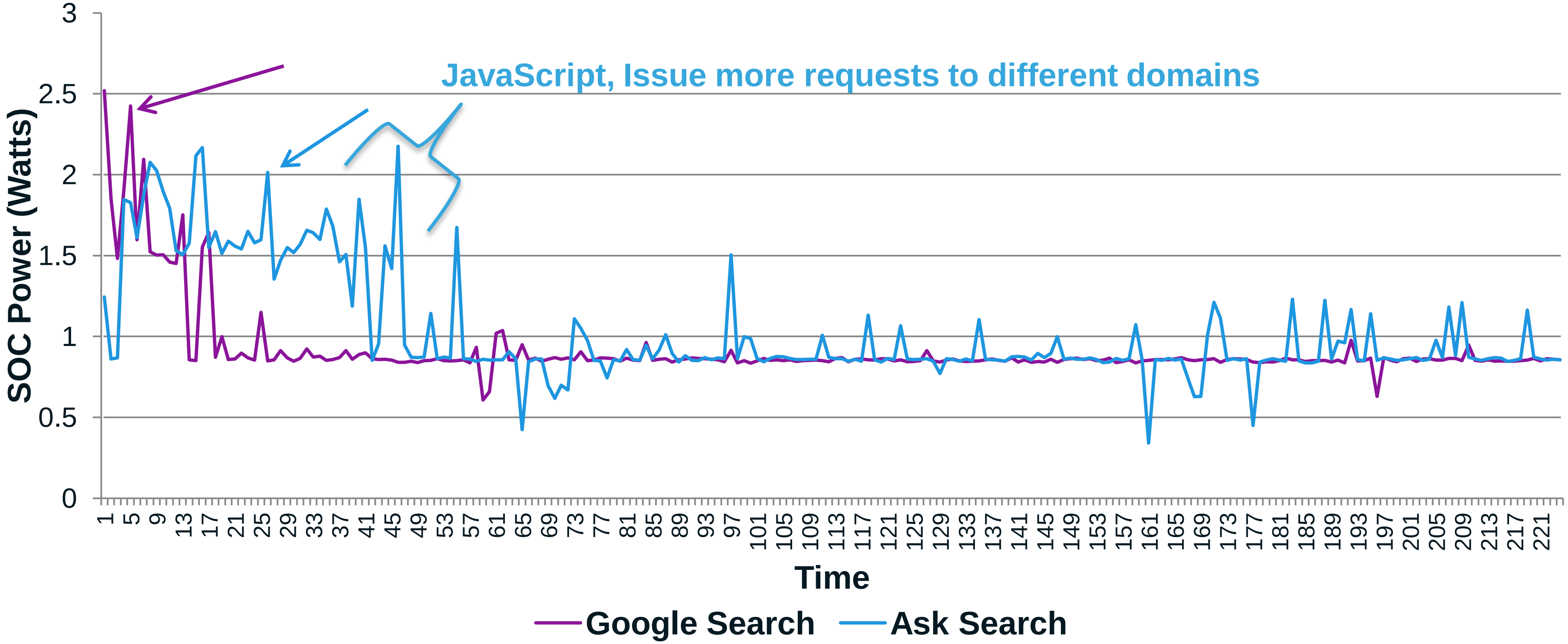
Tip 4:
Minimize Latency & JavaScript

Tip 4: Minimize Latency & JavaScript

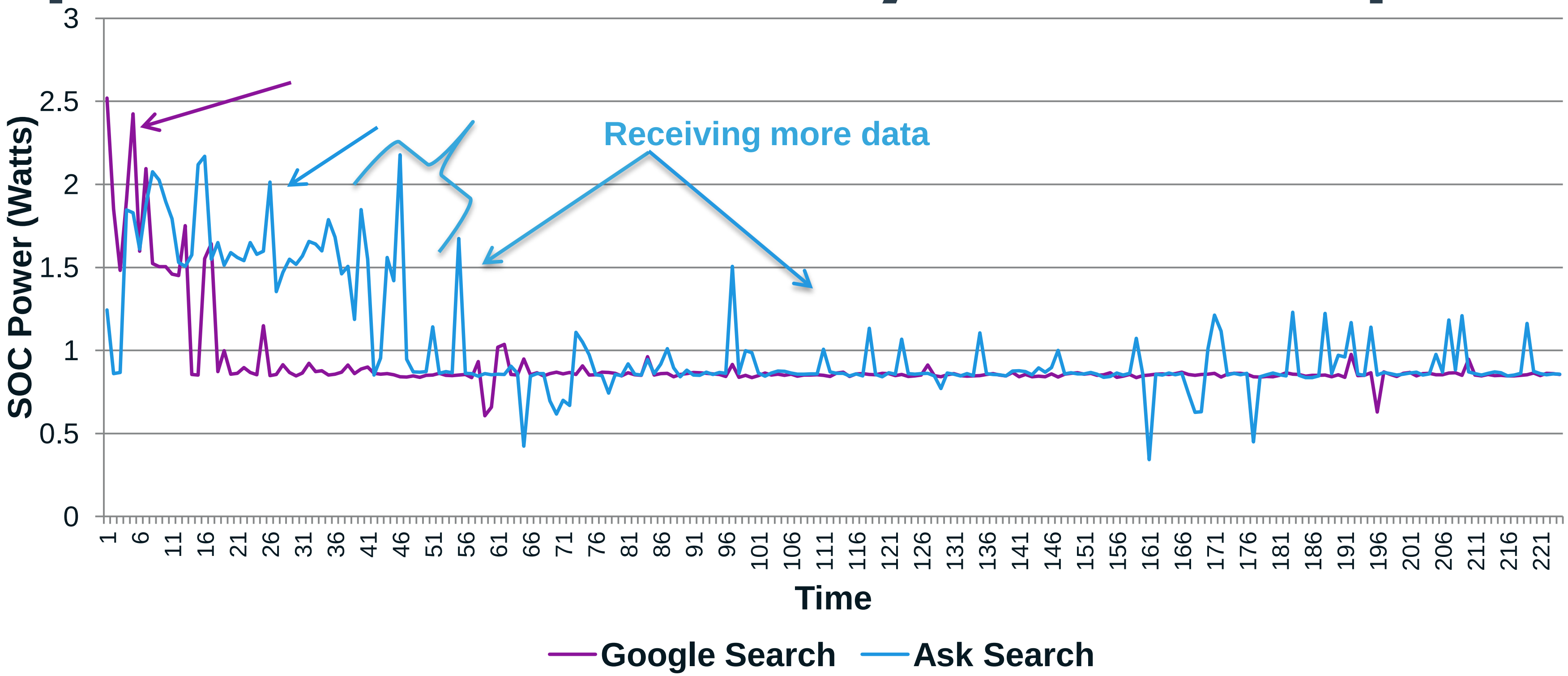


Average SOC power for Google search is 919mW, for Ask search, its 1031mW, ~10% difference

Tip 4: Minimize Latency & JavaScript



Tip 4: Minimize Latency & JavaScript



Tip 4: Minimize Latency & JavaScript

+ Batch requests
+ Cache external resources

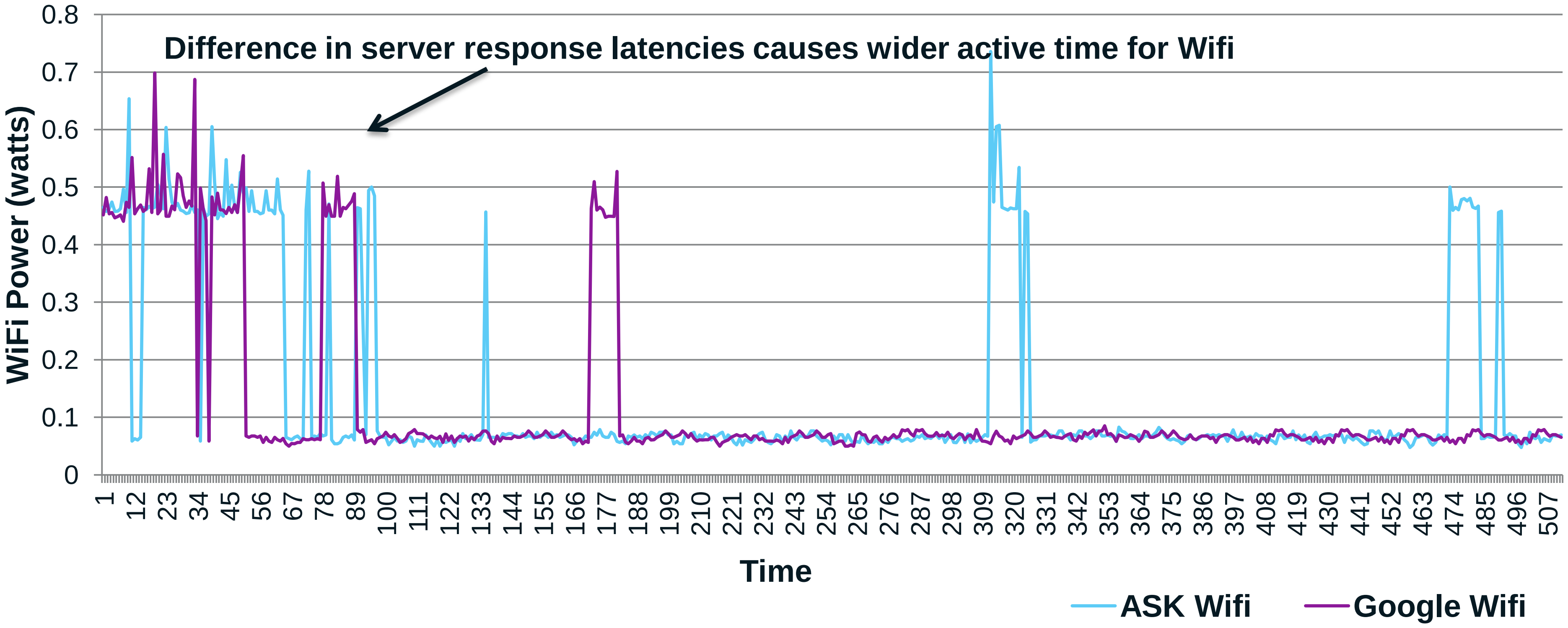
- Many sequential connections
- Long latencies

- Ask batches some requests, but not all.

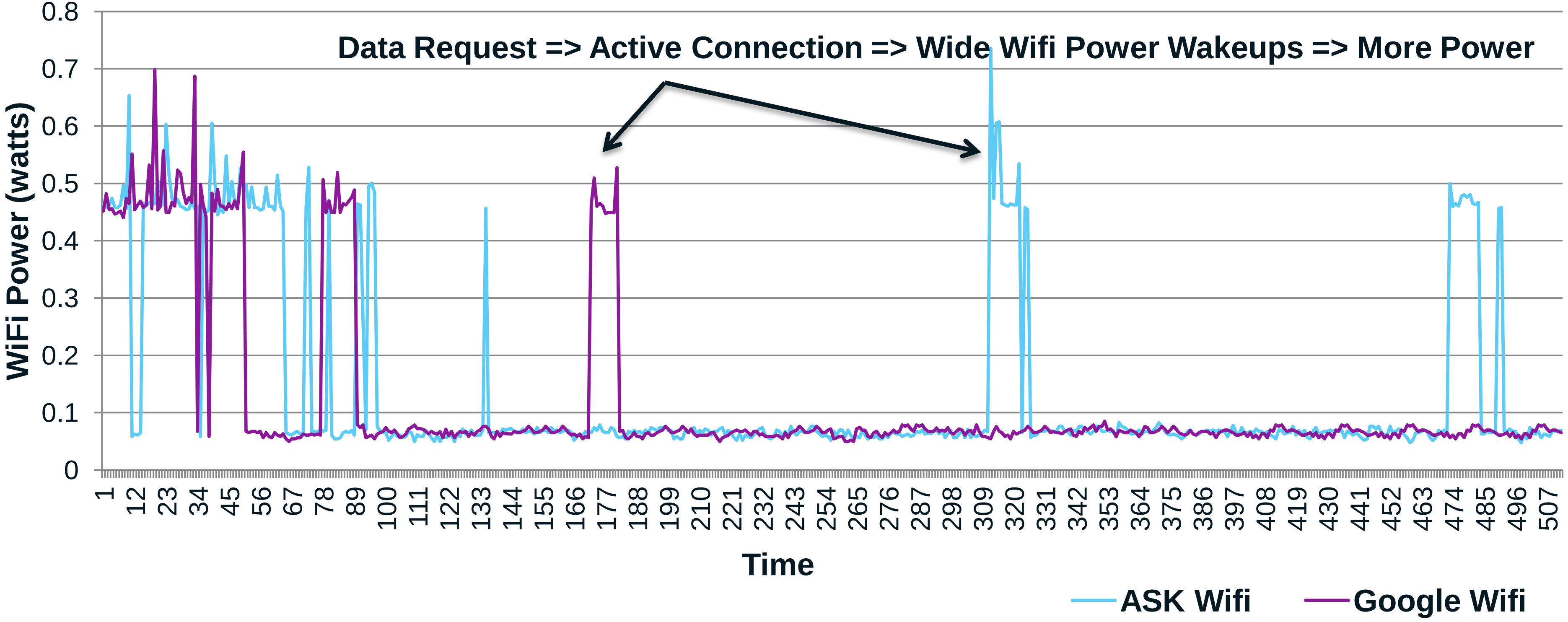
Web Applications

Tip 5:
Minimize Comm. Power

Tip 5: Minimize comm. power



Tip 5: Minimize comm. power



Data Request => Active Connection => Wide Wifi Power Wakeups => More Power

Tip 5: Minimize comm. power

- How can you reduce Wi-Fi power consumption?

  + Batch processing

  - Long latencies

Web Applications

Images are properties of their respective owners

Some other general guidelines for low power

- Context awareness
  - Minimize system resource use when not in foreground
    - Subscribe to screen and other system events
  - Free up temporary cache, files, and images
- Clean idle
  - Use resources only when user is active
  - Only act on user input
    - does piggybacking or deferring tasks to the next system wakeup work?
  - The power profile of an idle app should match system idle
  - Idle system CPU utilization target is less than 1%

Summary

- Battery life is an important selling factor
- Keep power in mind from Day 0 of development
- Follow our tips to make a good app-world citizen

Enjoy Developing Green Apps !

Email us at harita.chilukuri@intel.com & abhishek.dhanotia@intel.com

TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012