Designing an Energy Aware Operating System

Bringing the Smartphone Power Model to Windows PCs

ACM MobiCom 2013
October 2, 2013

Arun Kishan
Development Manager
Kernel Platform, Operating Systems Group
Microsoft Corporation

arunki@microsoft.com

Introducing Connected Standby

- Connected Standby (CS) is a screen-off low-power *active* state
  - Targeted at Intel and ARM System on Chip (SoC) systems

- “Smartphone Power Model”
  - Instant on/off
  - Extended standby battery life
  - Fresh content & real-time notification handling

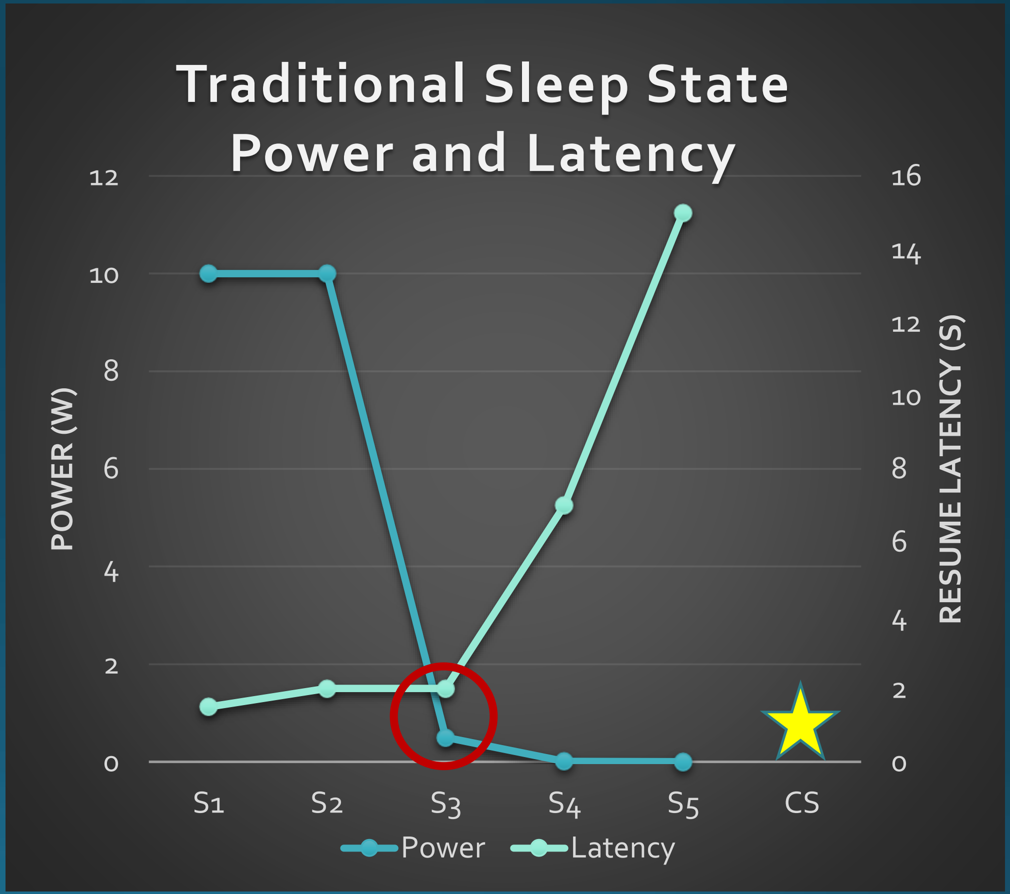
- Supported platforms include:
  - All Windows RT ARM tablets
  - Intel Clovertrail tablets
  - Next-generation Intel Haswell ultrabooks and Baytrail tablets

What about Standby and Hibernate?

- Traditional PCs support multiple power states
  - Standby: *Suspend-to-RAM / S3*
  - Hibernate: *Suspend-to-Disk / S4*
  - Off: *Shutdown / S5*

- Globally-Coordinated Transitions
  - All hardware and software components are either on or off

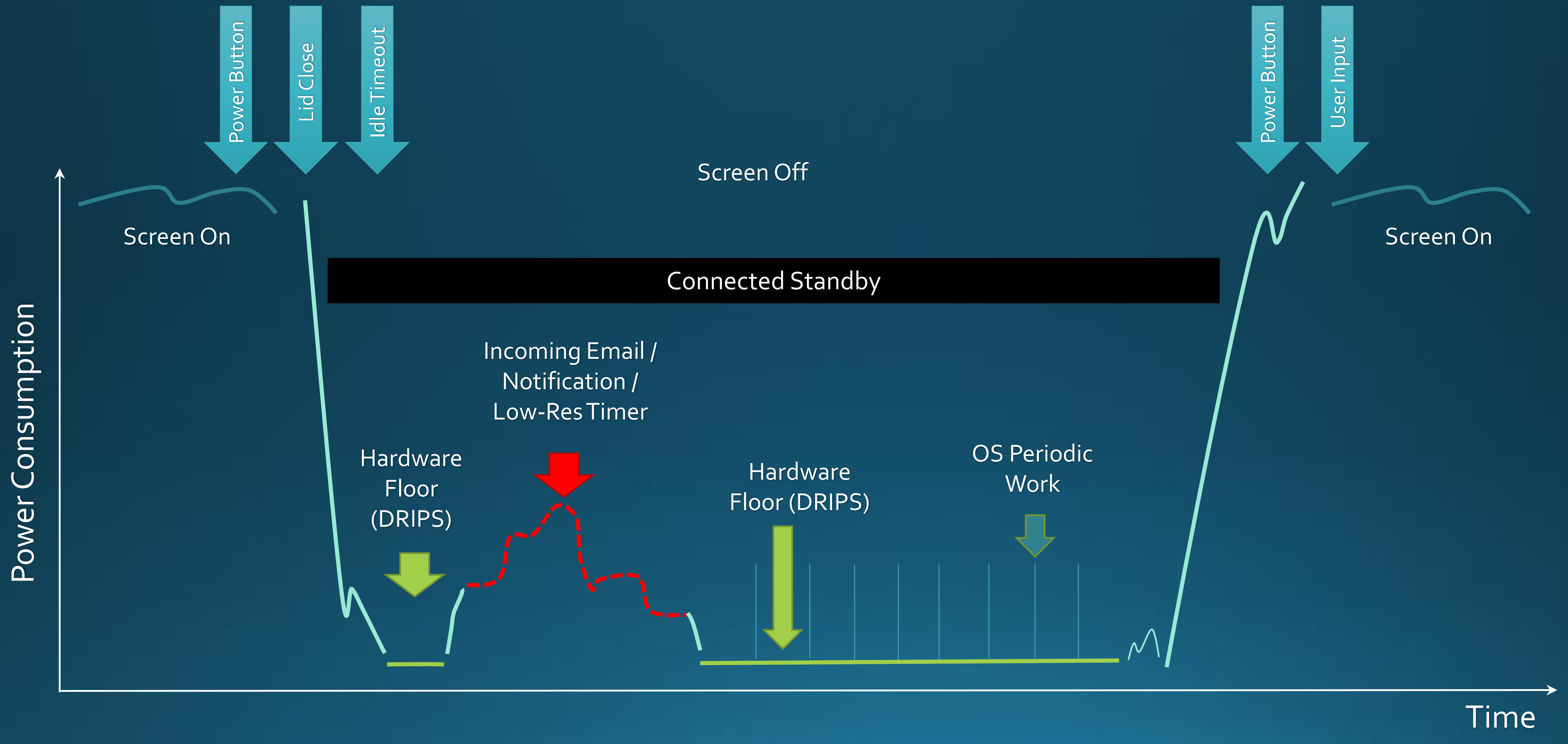
- Problems:
  - Tradeoff power and on/off latency
  - No network connectivity when "off"
  - "Partially" active states not possible



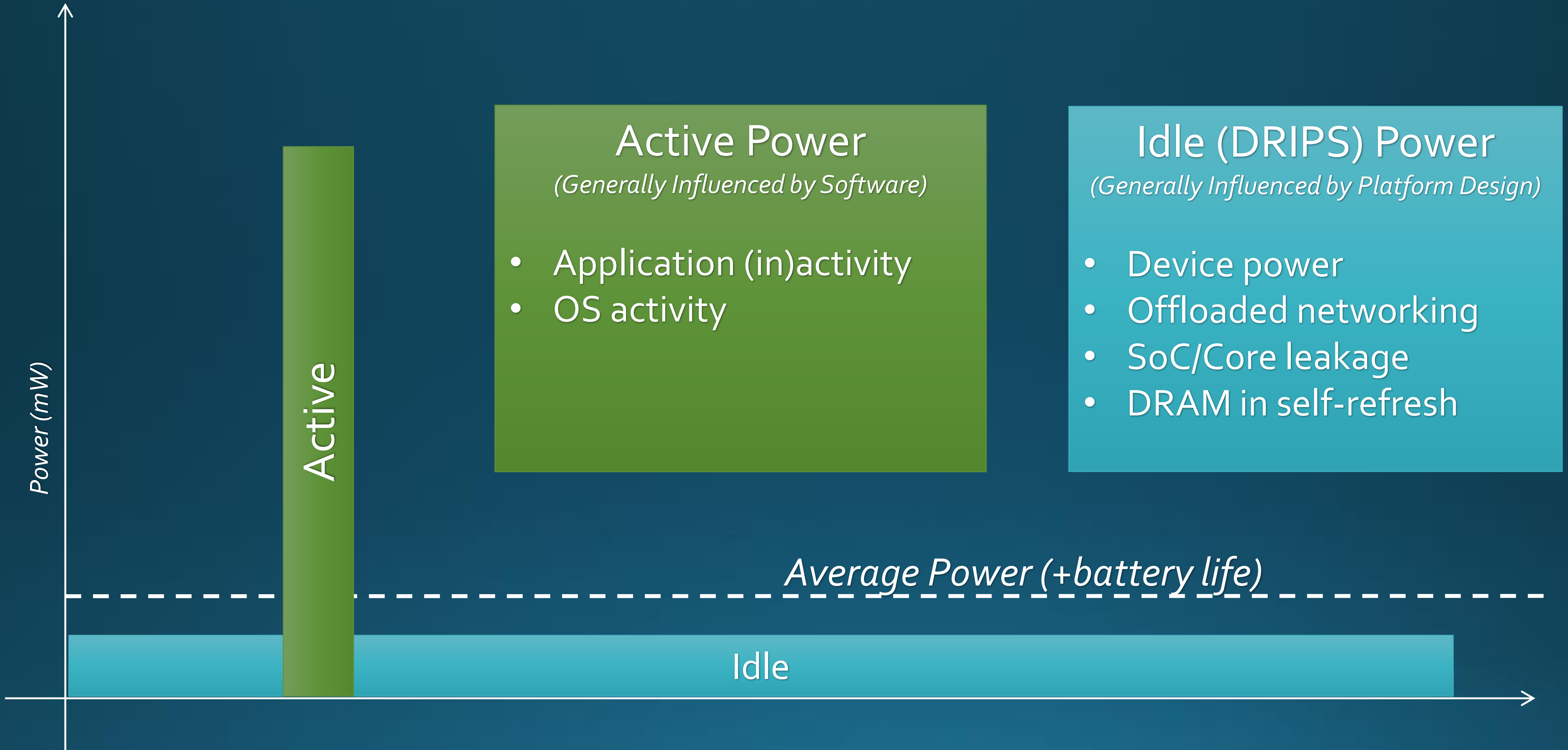
Design Goals for CS Systems

- Windows devices and Store applications are able to provide an always-on, always-connected experience
- 14 days of CS time on a full battery charge for the typical system design
- Devices continue to deliver exceptional battery life regardless of the number of Store applications
- Maintain compatibility for existing Win32 applications

A Connected Standby Session



Active, Idle, and Average Power in CS

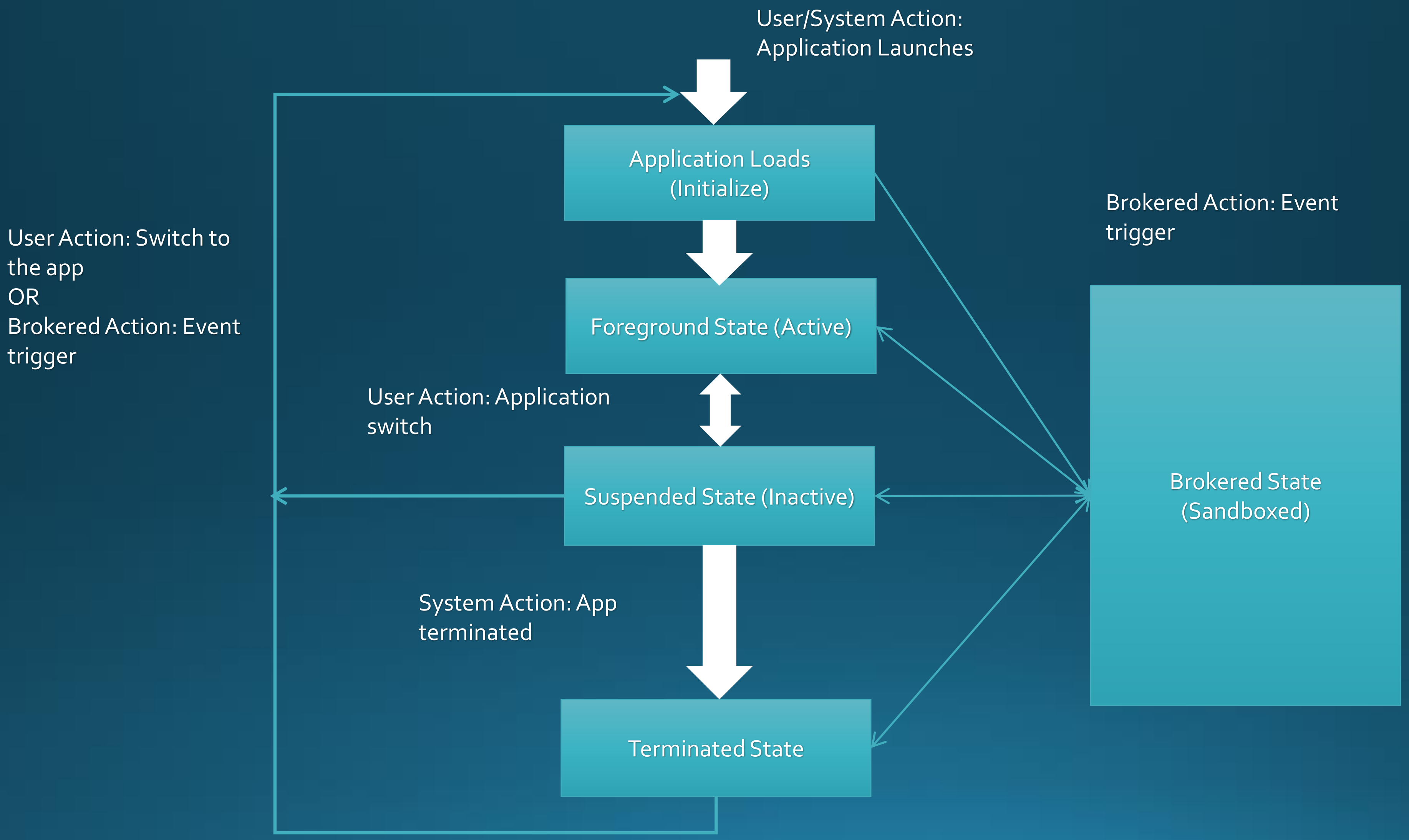


Long Battery Life = Maximizing DRIPS residency

Challenges with Connected Standby

- **Background execution**
  - **Enabling multi-tasking scenarios while preserving battery life**
- Device power management
  - Devices and SoC must enter low-power mode
- Network Connectivity
  - Maintain connectivity even while the SoC is off
- Staying Idle
  - Confine periodic tasks performed by the OS

Windows 8 Execution Model



Background Tasks

- *Brokered background tasks* are the only way to achieve arbitrary background execution

- Applications communicate with system *brokers* to arrange for brokered events
  - Two types of events: *triggers* and *conditions*
  - Brokers: Time, Real-Time Communication (RTC), Windows Notification Service (WNS), and System Events
  - Events: 15-minute timer, remote wake, keep-alive, internet available, geofence, etc.

- Applications associate a *background task* with a set of events
  - Applications create a logical expression using a trigger and zero or more conditions
  - Example: Run mail download every 15 minutes, but only when the internet is available

Controlling Background Execution

- Sandbox policy consists of multiple elements
  - Limited CPU cycles and network time [battery preservation]
  - Reduced CPU, I/O, and memory priority [foreground responsiveness]

- Sandbox policy only applies when the application is in the background state

- Applications are classified by the user as *lock screen* capable or not
  - Lock screen is designed to show status for always on, always connected applications (7 applications maximum)
  - Limited quota for background resources (2 CPU seconds and 4 network seconds every 15 minutes)
  - Non-lock screen applications generally *cannot* execute in the background

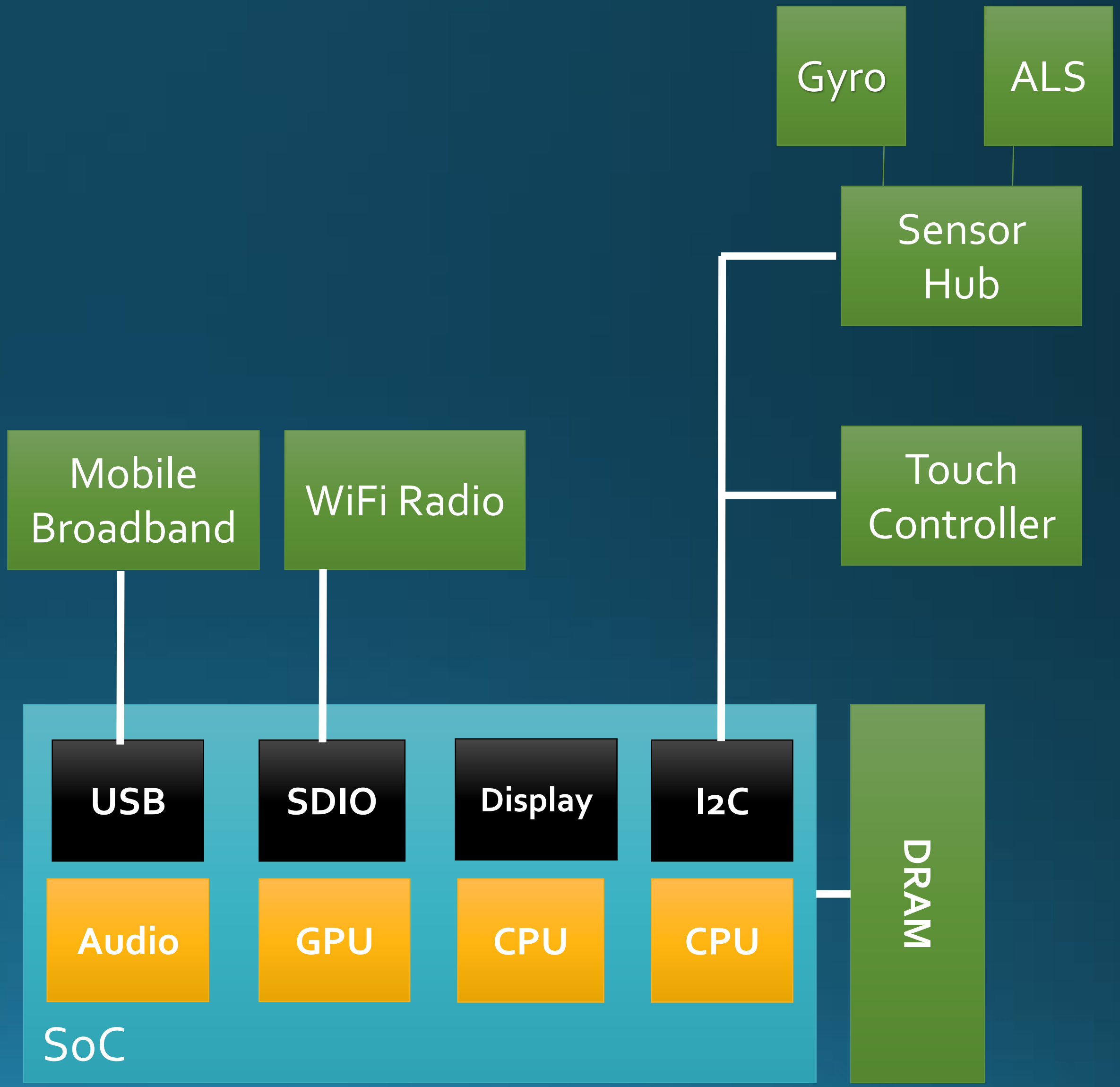
Challenges with Connected Standby

- Background execution
  - Enabling multi-tasking scenarios while preserving battery life
- **Device power management**
  - **Devices and SoC must enter low-power mode**
- Network Connectivity
  - Maintain connectivity even while the SoC is off
- Staying Idle
  - Confine periodic tasks performed by the OS

Device Power Management

- DRIPS (*deepest resident platform idle state*) requires:
  - All CPUs in deep idle
  - On-SoC device state saved
  - Platform (off-SoC) devices in low-power mode
  - DRAM is in self-refresh
  - Network in offload mode

- All devices should participate in runtime power management (PoFx)
  - Monitor and track application interaction (active handles, operation count, etc.)
  - Runtime idle to D3 (off) when not in use

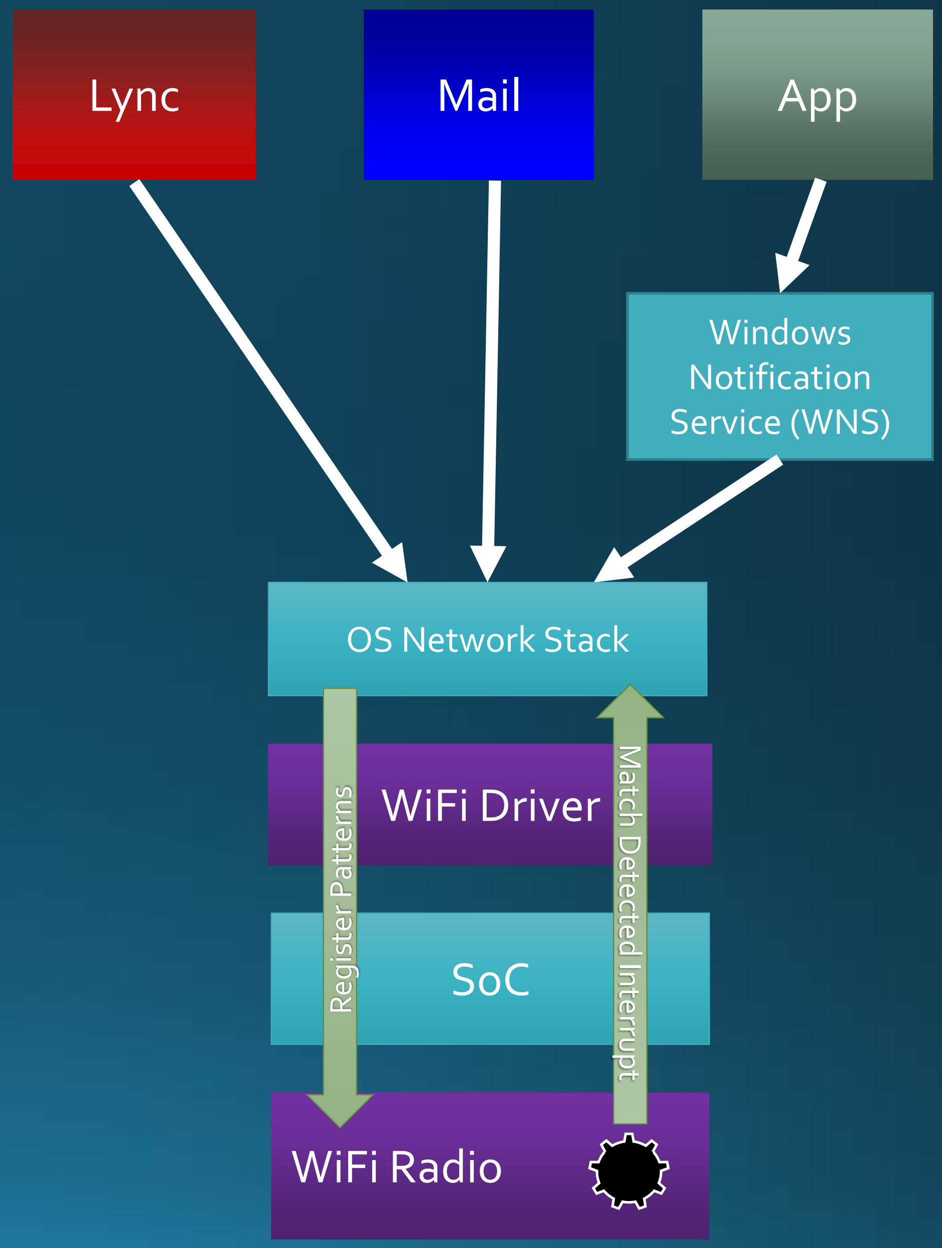


Challenges with Connected Standby

- Background execution
  - Enabling multi-tasking scenarios while preserving battery life
- Device power management
  - Devices and SoC must enter low-power mode
- **Network Connectivity**
  - **Maintain connectivity even while the SoC is off**
- Staying Idle
  - Confine periodic tasks performed by the OS

Offloading Connectivity

- CS depends on the network adapter to efficiently maintain connectivity through *offloads*
  - ARP/NS protocols
  - Network List Offload (NLO)
  - Wireless GTK rekey
- Windows can register a set of "wake patterns" on the adapter
  - Used by RTC and WNS brokers
  - When matched, CPU will be interrupted
- Network remains connected at all times!
  - No reconnect as in wake/resume from S3/S4



Challenges with Connected Standby

- Background execution
  - Enabling multi-tasking scenarios while preserving battery life
- Device power management
  - Devices and SoC must enter low-power mode
- Network Connectivity
  - Maintain connectivity even while the SoC is off
- **Staying Idle**
  - **Confine periodic tasks performed by the OS**

Resiliency Technologies

- Despite revamped application model, device power management, and network offload, periodic Windows system activity can prevent DRIPS

- *Resiliency* dampens and coalesces activity to boost DRIPS residency

- Resiliency technologies consist of multiple telescoping tiers
  - *Desktop Activity Moderator (DAM)* reduces activity from traditional services and applications
  - *Network Quiet Mode (NQM)* reduces power and limits network software stack activity
  - *I/O Coalescing* caches I/Os from registry, cache manager, etc. in RAM to prevent disk access

- System behavior is not deterministic when resiliency technologies are engaged
  - OS manages requests from system components to turn off resiliency tiers as appropriate
  - Example: NQM is disabled by the OS when executing background tasks that require network

Achieving Exceptional Battery Life

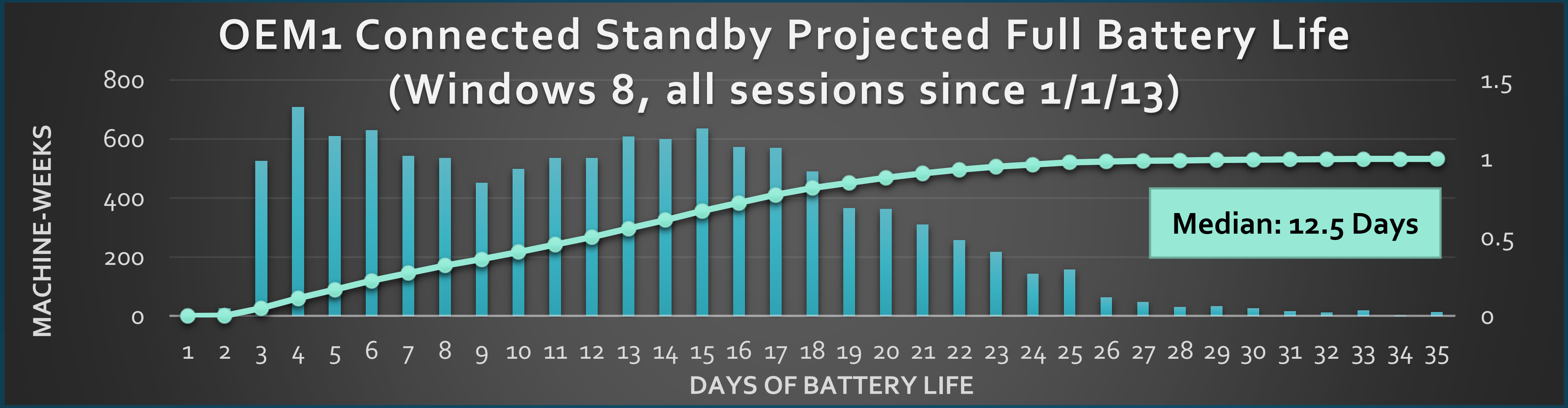
- Battery life dependent on how long the system can stay in DRIPS

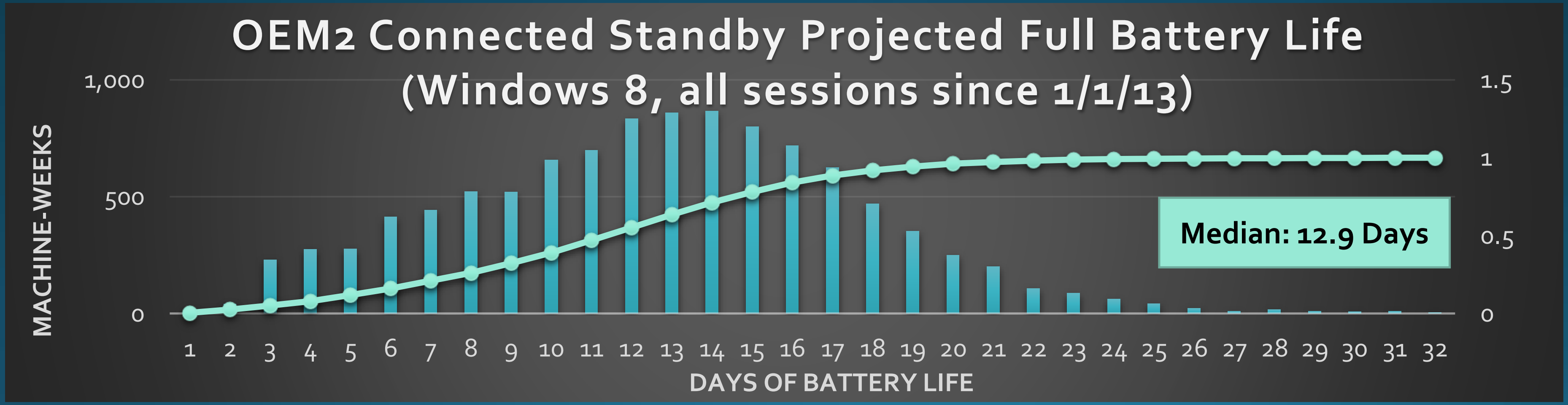
- Multiple inputs feed into platform idle state selection
  - *System latency*: User responsiveness requirement (15ms in active, 150ms in CS)
  - *Idle residency*: Expected idle time for CPUs (function of the hardware interrupt frequency)
  - *Device state*: Power state of the devices (D0 – D3)

- Entering CS lowers latency requirement (user no longer present), increases idle residency (applications suspended), and enables devices to power down

- Kernel consumes the above information and selects the appropriate platform idle state
  - Examples: NV LP0, QC PXO Shutdown, Intel S0i3
  - SoC vendor's *Power Engine Plugin* (PEP) executes actual state transition
  - System ultimately enters DRIPS

Results


OEM1 Connected Standby Projected Full Battery Life
(Windows 8, all sessions since 1/1/13)
MACHINE-WEEKS
800
600
400
200
0
1.5
1
0.5
0
Median: 12.5 Days
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
DAYS OF BATTERY LIFE


OEM2 Connected Standby Projected Full Battery Life
(Windows 8, all sessions since 1/1/13)
MACHINE-WEEKS
1,000
500
0
1.5
1
0.5
0
Median: 12.9 Days
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
DAYS OF BATTERY LIFE

Future Directions

- Expanding CS to more form factors
  - Ultrabooks, desktops, servers, all-in-ones (AIOs)
- Reducing hardware cost requirements on CS
  - Replace SSDs with hybrid rotational disks
  - Replace LP-DDR with memory power management
- Improve fidelity of background resource policy
  - Expand energy resource accounting to include disk, GPS, etc.
- Shift to *adaptive* application power model
  - Allow an unbounded number of applications to run in the background
  - Accurately measure application power usage online
  - Only throttle applications with unexpected energy consumption

Questions?

Arun Kishan
Development Manager
Kernel Platform, Operating Systems Group
Microsoft Corporation

arunki@microsoft.com