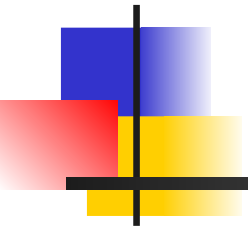
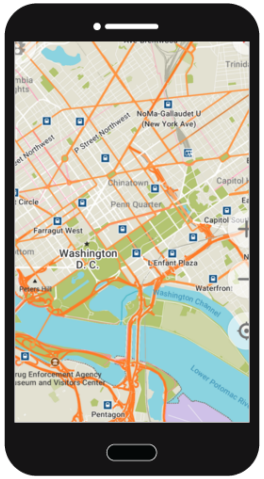


CS476/576: Programming Models for Emerging Platforms

LAB: A Programming Model for Mobile Sensing

Context Sensing

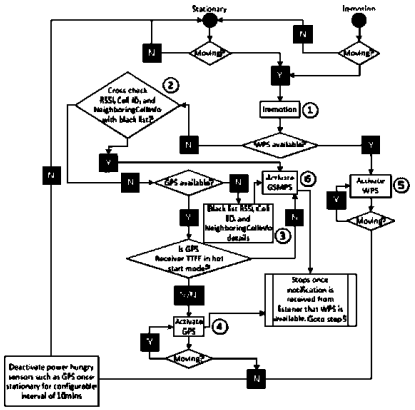


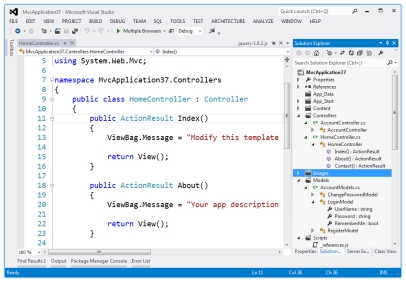
Existing Programming Models

Context = {location, activity, social, mood, ….}

But continuous context sensing drains the battery

Solution?



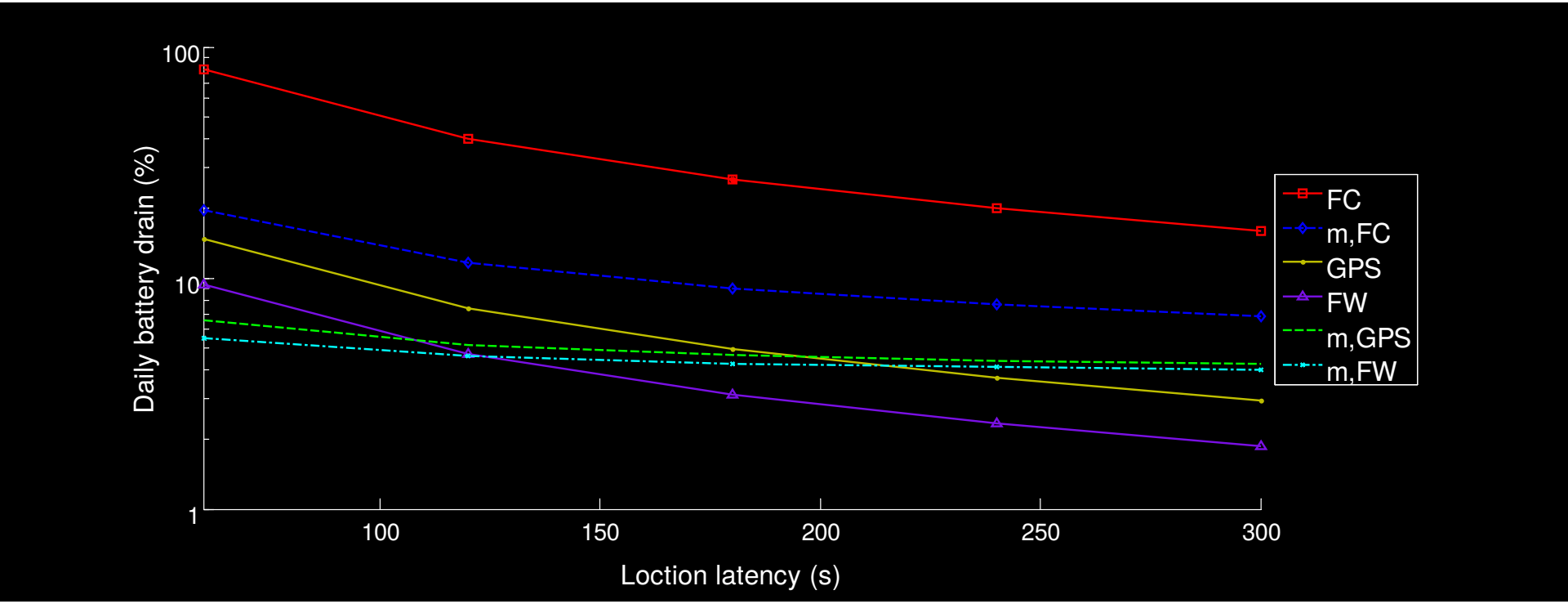


Developer carefully studies and selects efficient algorithms

Developer tunes for battery drain

Developers are only human

Optimal location tracking algorithm changes with latency

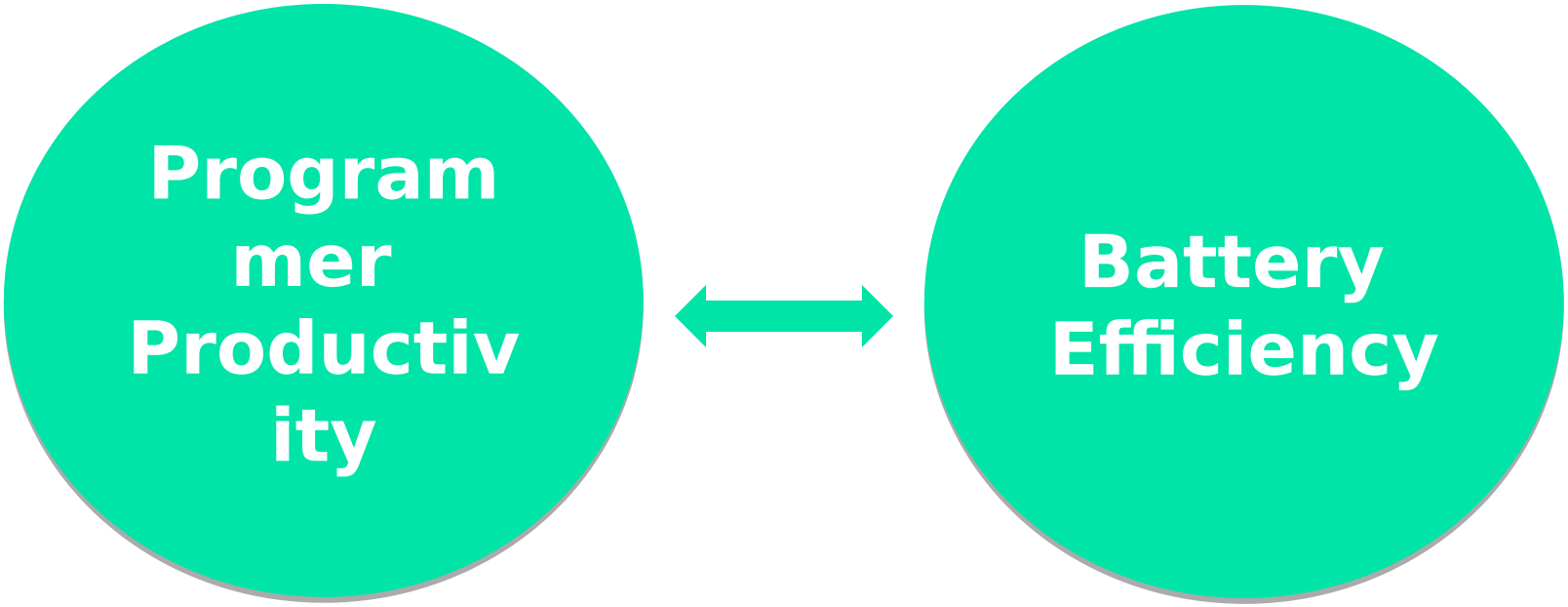


Existing OS APIs

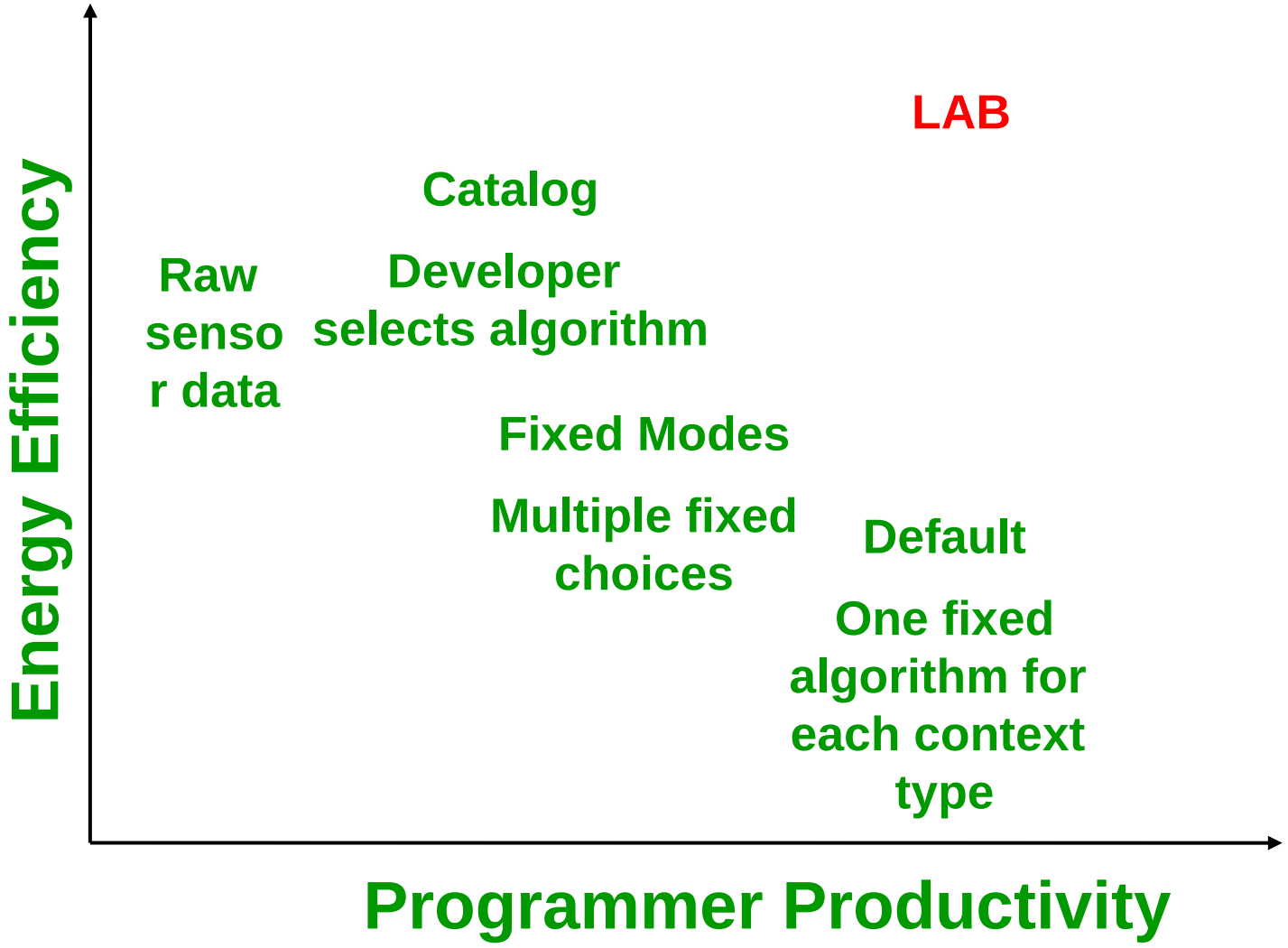
- Android: `addProximityAlert`
- Standby life reduced from 430hrs to

Phone	Standby Life if API used (hrs)
HTC Desire S	12
Galaxy Nexus	19
HTC HD2	5

Challenge



Choices for API



Design Goal 1

Programmability

Get it working

Carefully optimize

- Simple to sophisticated programmers
- OS algorithms evolve with technology
  - Expose “logical context”, not technology, to developers

Design Goal 2

OS have sufficient information to select most efficient algorithm and parameters

Insight

Latency, Accuracy, and Battery

- Priority order expresses rich semantics and is easy to program
  - e.g. Latency more important than battery
- Quantitative parameters add expressiveness, but are not easy to guarantee
  - e.g. Latency < 60s

API

`ChangeAlert`

- Context[]: collection of locations or activities to detect
- FirstPriority: one of accuracy, latency, or battery
- Value: quantitative constraint for first dimension (such as percentage of battery drain per 24 hours)
- SecondPriority: one of accuracy, latency, or battery
- Value: quantitative constraint for second dimension
- ThirdPriority: one of accuracy, latency, or battery
- Value: quantitative constraint for third dimension

Examples

Detect when user starts walking

Activity[] w = {Activity.WALK}

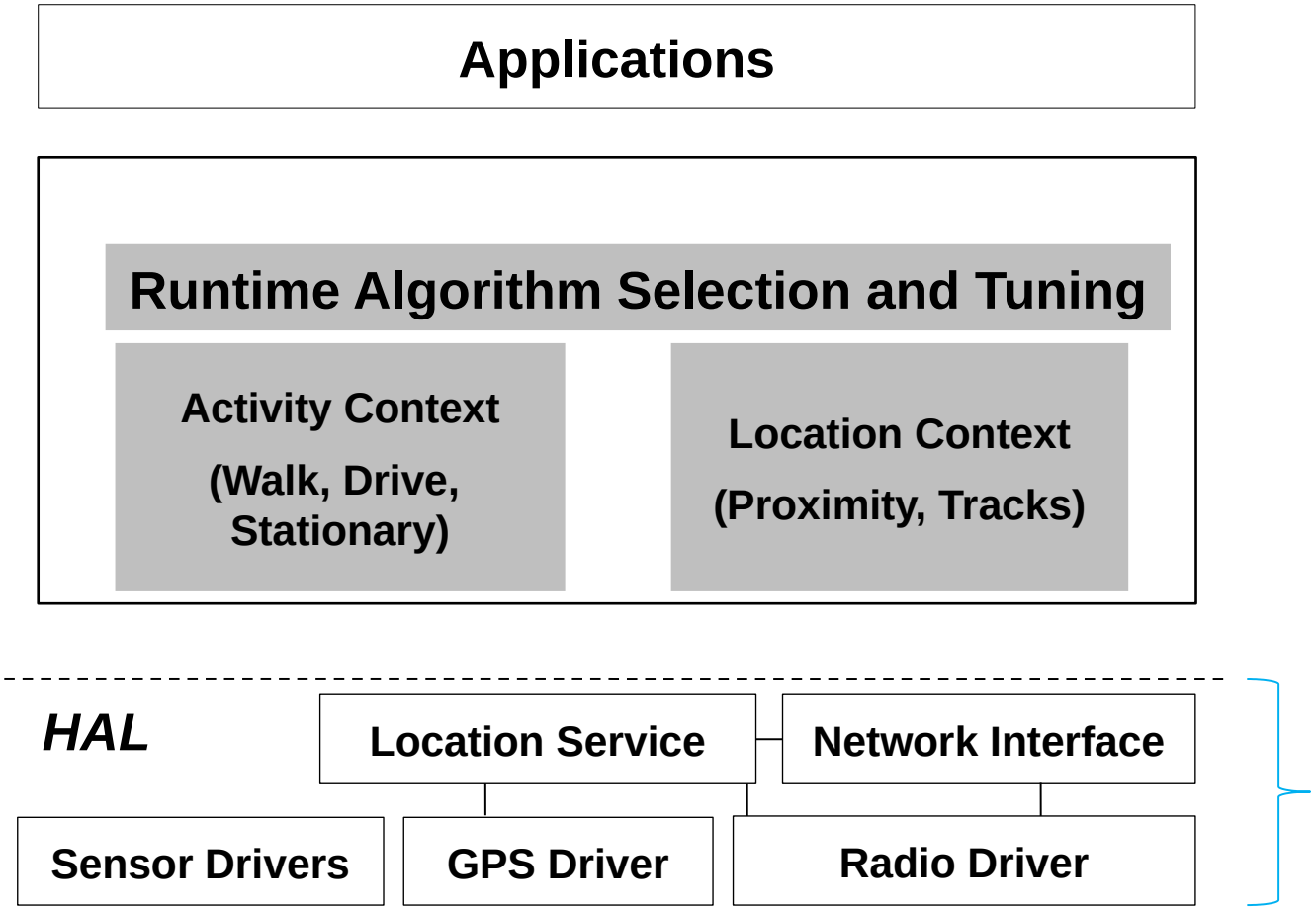
ChangeAlert(w)

Count driving trips

Activity[] d = {Activity.DRIVE}

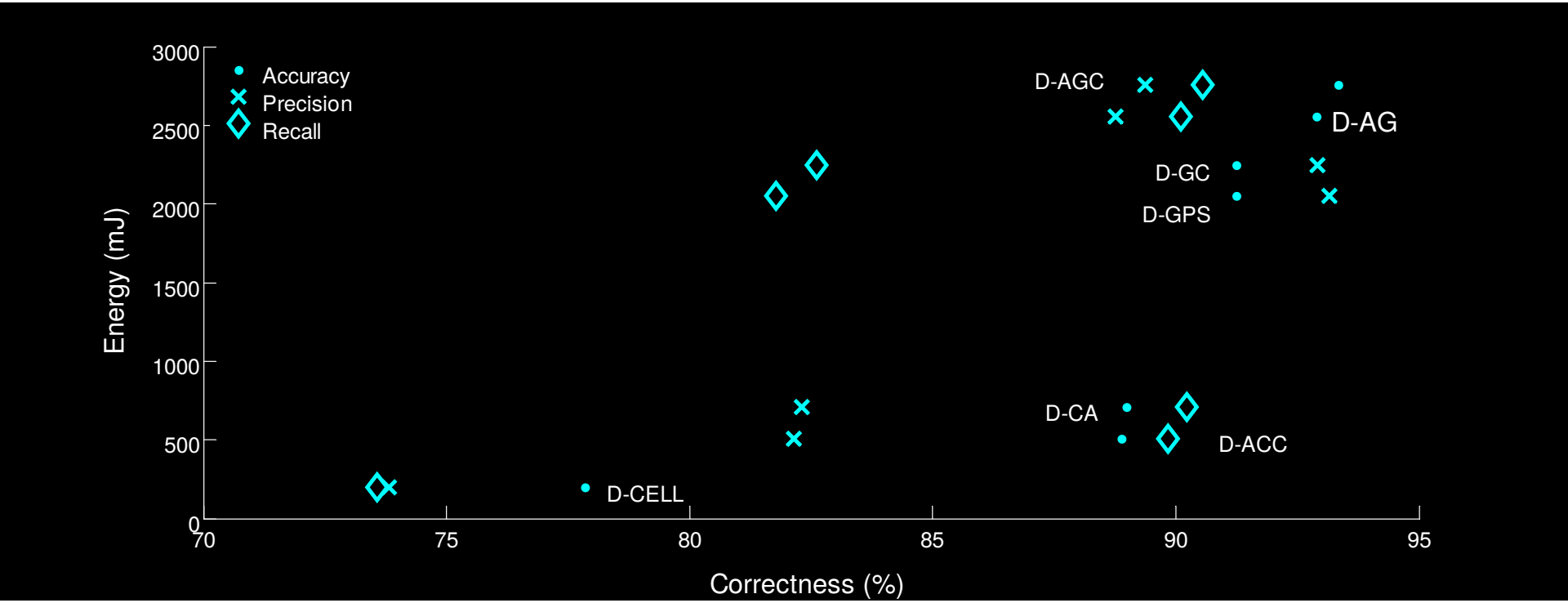
ChangeAlert(d, Priority.Battery, 1%, Priority.Latency, 300s)

Implementation



Driving (7 algorithms)

Accuracy vs. Energy

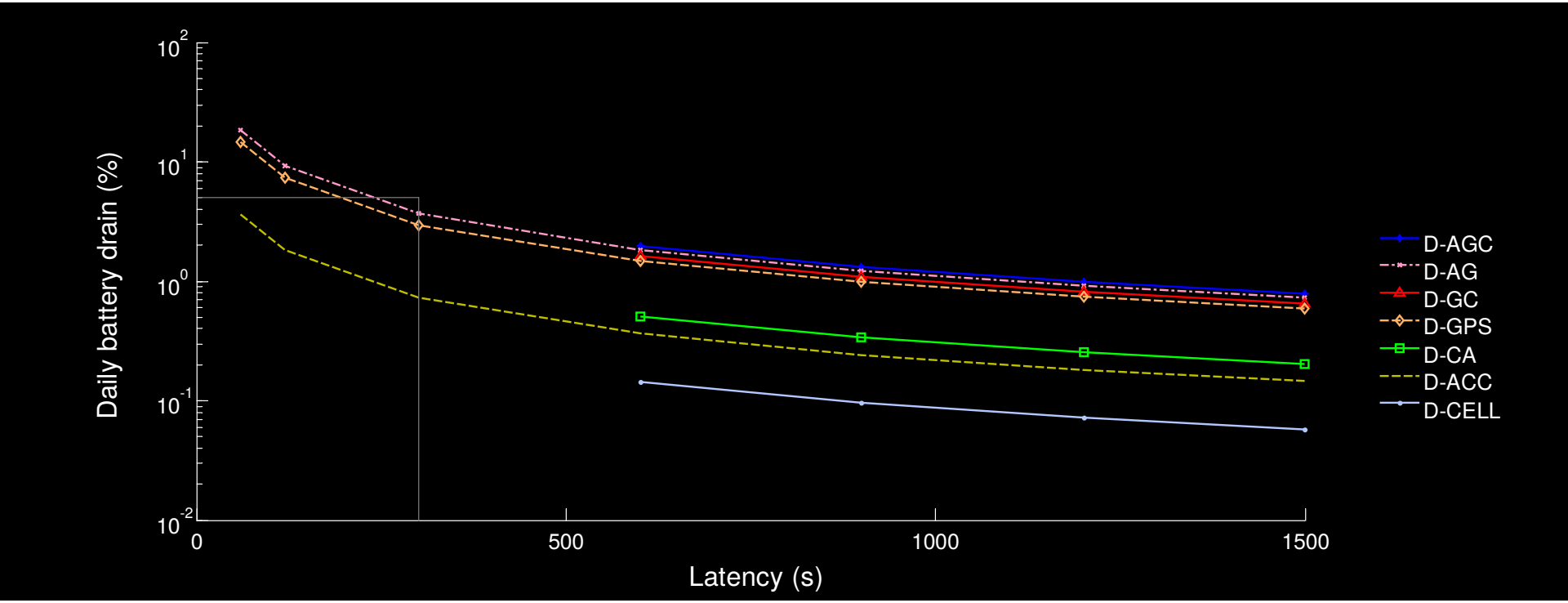


- Accuracy alone can be misleading: when users drive only a small fraction of the time, an algorithm that always outputs not-driving will be accurate

precision (false positive): correct report/all reported

Recall (fall negative): correct report/correct all

Driving (7 algorithms), contd.

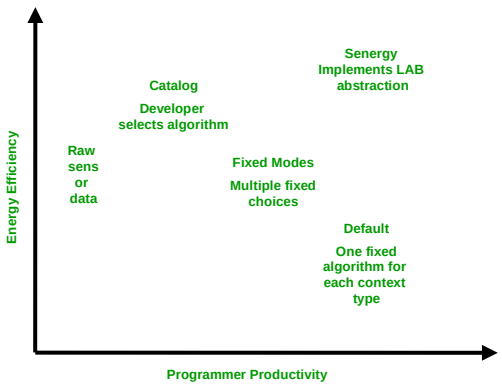


Latency and Energy

Computed trade-off data for all our contexts algorithms (driving, walking, stationary, location, proximity)

LAB vs. Other API Options

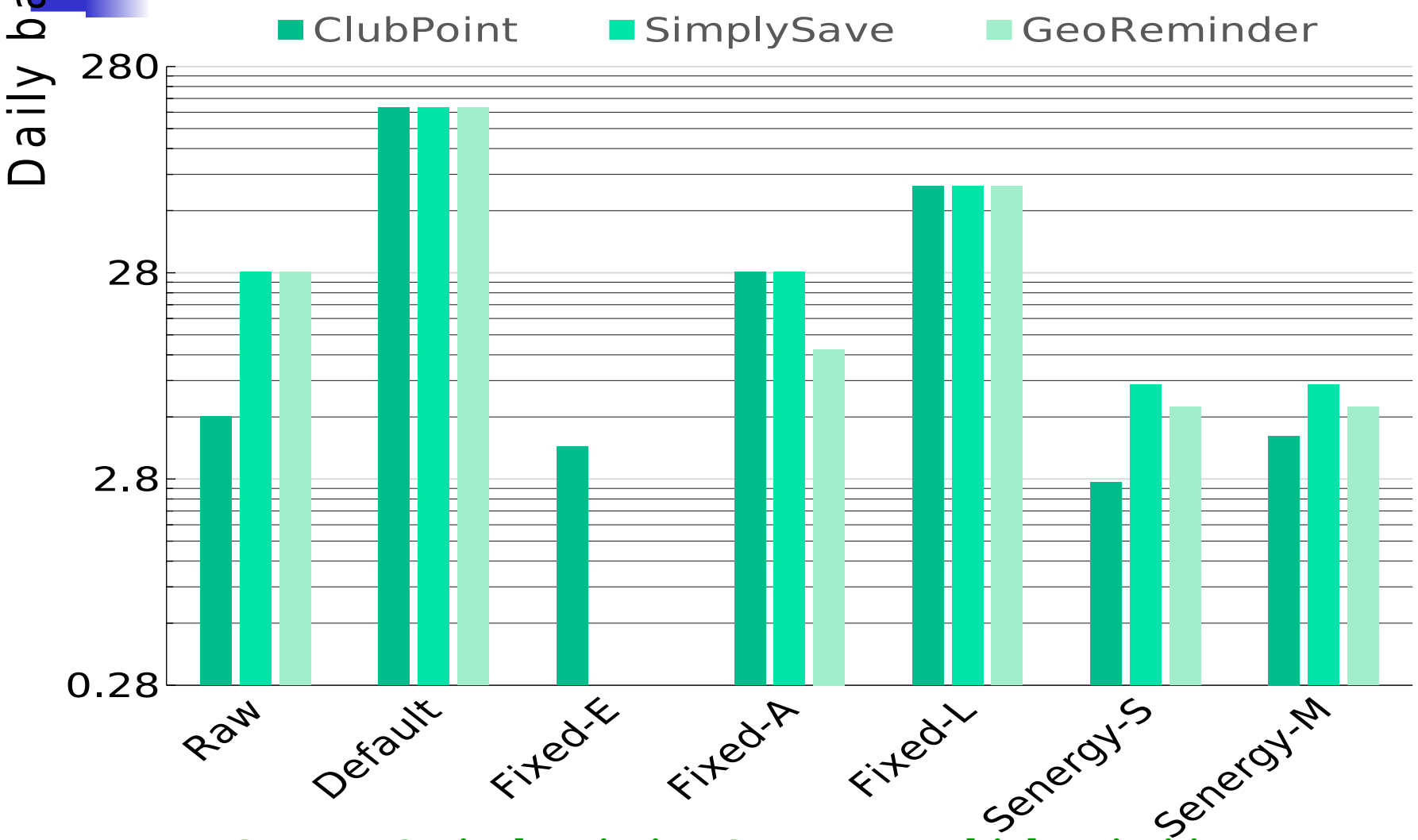
- LAB selects algorithm at run time based on API parameters
- Evaluated on 4200 user-hours of mobile sensor data (49 unique users)
- Compare against
  - Raw (developer does context inference, e.g. lower-level Android programming)
  - Default (OS chooses one default)
  - Modes, Eg: Energy mode, Accuracy mode, Battery mode



Location Apps

- ClubPoint
  - Remind me to use the AAA card if at a participating location for long enough buy
  - ChangeAlert(…, Priority.Latency, 300s)
  - ChangeAlert(…, Priority.Latency, 300s, Priority.Battery, 5%, Priority.Accuracy)
- SimplySave
  - Remind if near any participating location with coupon, fast
- GeoReminder
  - Remind if near a specific (single) location

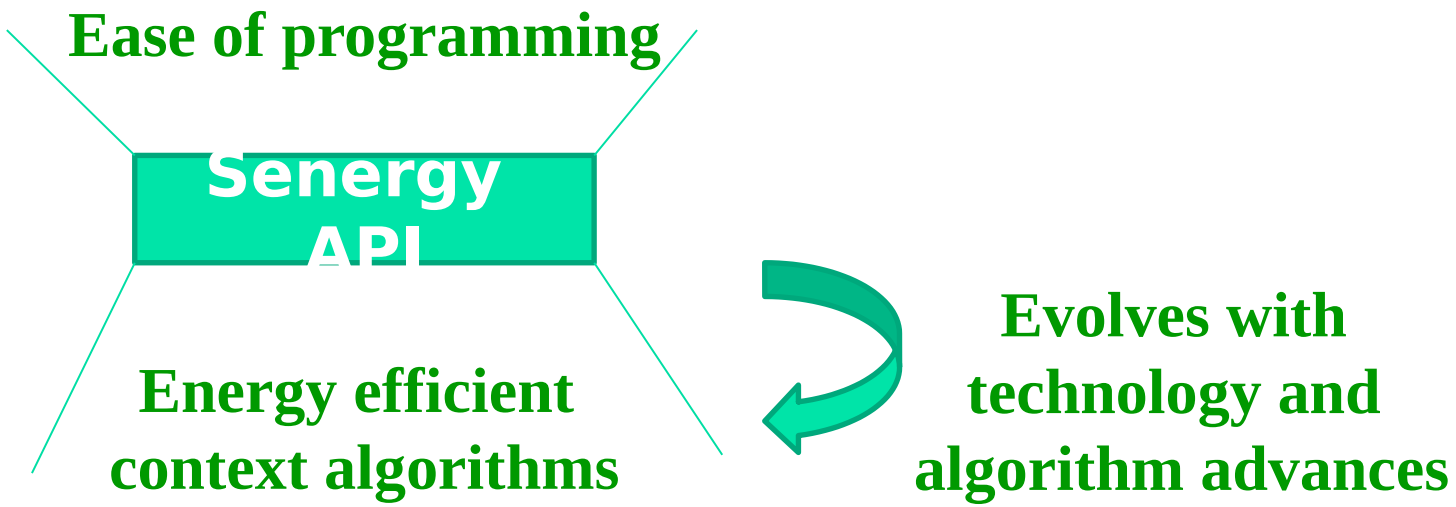
Results (SENERGY uses LAB)



Senergy-S: single priority; Senergy-: multiple priorities

Conclusions

- Selecting the right algorithms and tuning their energy use is hard
- Latency, Accuracy, and Battery capture context trade-offs



Acknowledgments

- Jules White, Android Sensors and Callbacks
- Amin Kansal, et. al, The Latency, Accuracy, and Battery (LAB) Abstraction: Programmer Productivity and Energy Efficiency for Continuous Mobile Context Sensing