Beam: Ending Monolithic Applications for Connected Devices

Chenguang Shen *(UCLA)*

Rayman Preet Singh *(Univ. of Waterloo/Samsung Research)*

Amar Phanishayee, Aman Kansal, Ratul Mahajan *(Microsoft Research)*

Growth in Connected Devices

Internet of Things (IoT)

# of sensing devices > # of people since 2008

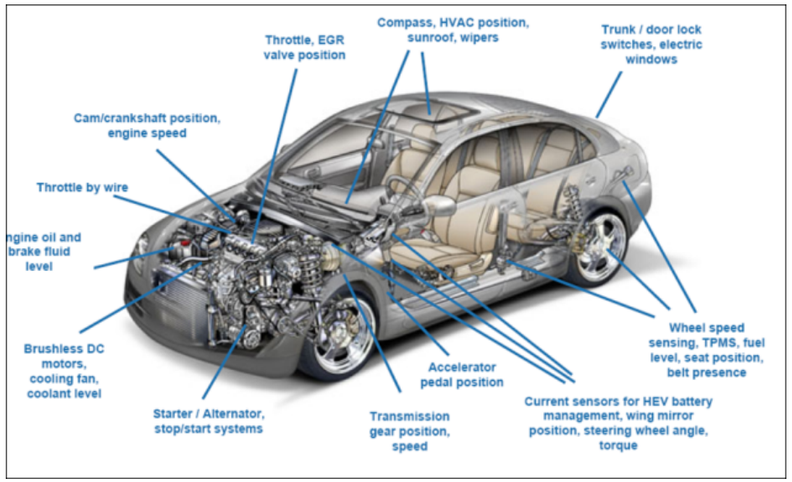
50 billion *connected sensing devices* by 2020

http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

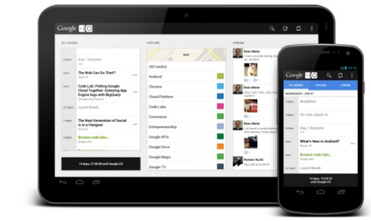
Growth in Connected Devices

Smart Home Devices

Sensors in Commercial Spaces



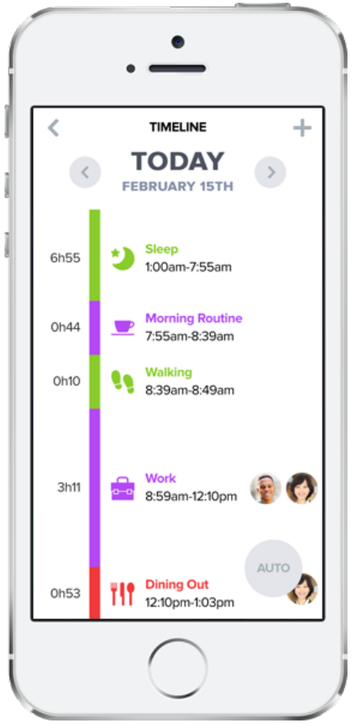
Automobile Sensors

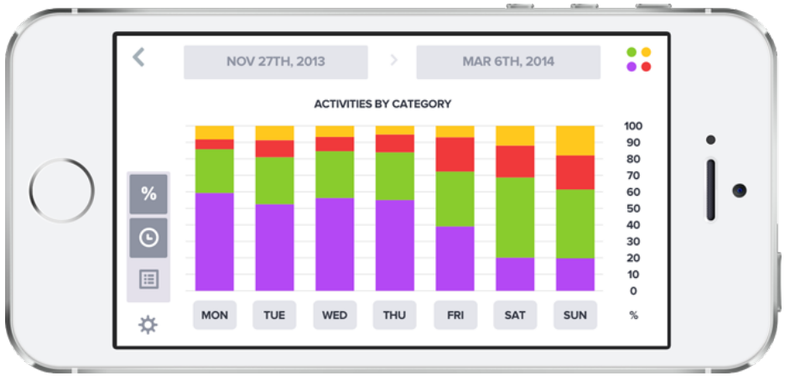


Personal Devices

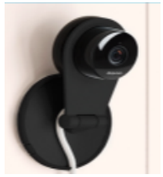
mHealth Devices

Sample App: Quantified Self





Quantified Self App



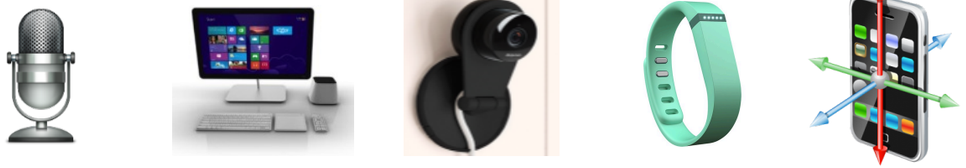
Challenges in Developing IoT Apps

Tight Coupling

App-device Silos

- ***Challenge 1: development effort***
  - Device driver
  - Inference logic
  - User interface
  - Cloud service

Quantified Self App



Challenges in Developing IoT Apps

Heterogeneous
Hardware Devices

- Camera

- Fitness Activity

- ***Challenge 2: device selection***
  - Discover devices in a deployment
  - Select appropriate devices from deployment
  - Support settings with user mobility where available devices might change

Quantified Self App

Challenges in Developing IoT Apps

Mobile and
Geo-distributed Devices

- ***Challenge 3: disconnection tolerance***
  - App should work even with network disconnections

Quantified Self App (e.g., Fitness Activity Tracking)

Challenges in Developing IoT Apps

Battery-powered
Mobile Devices

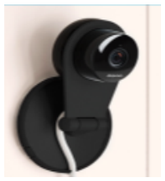
- ***Challenge 4: efficient resource usage***
  - Efficiently partition computation across available devices

Quantified Self App









Recap of Key Challenges

- Development effort
- Device discovery and selection
- Disconnection tolerance
- Efficient resource usage

Beam Overview

Quantified Self App

Subscribe(FitnessActivity, params)

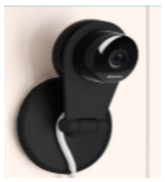
Beam: programming abstraction + associated runtime

- Insight: decouple *what is sensed and inferred* from *how it is sensed and inferred*
  - Raise the abstraction from data to **inferences**
- Key abstraction: ***inference graph***
  - Simplifies development, enables device selection, support device disconnections





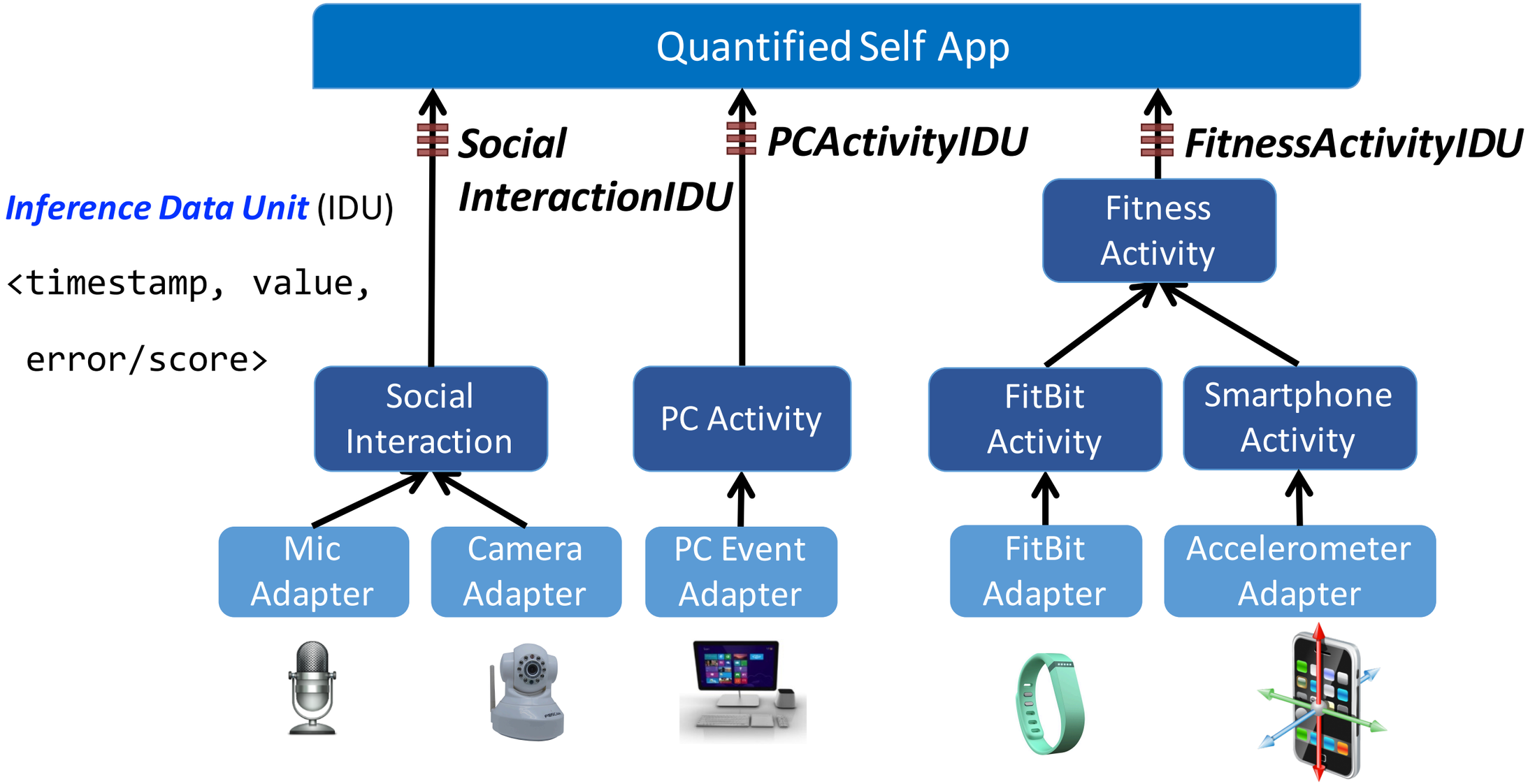




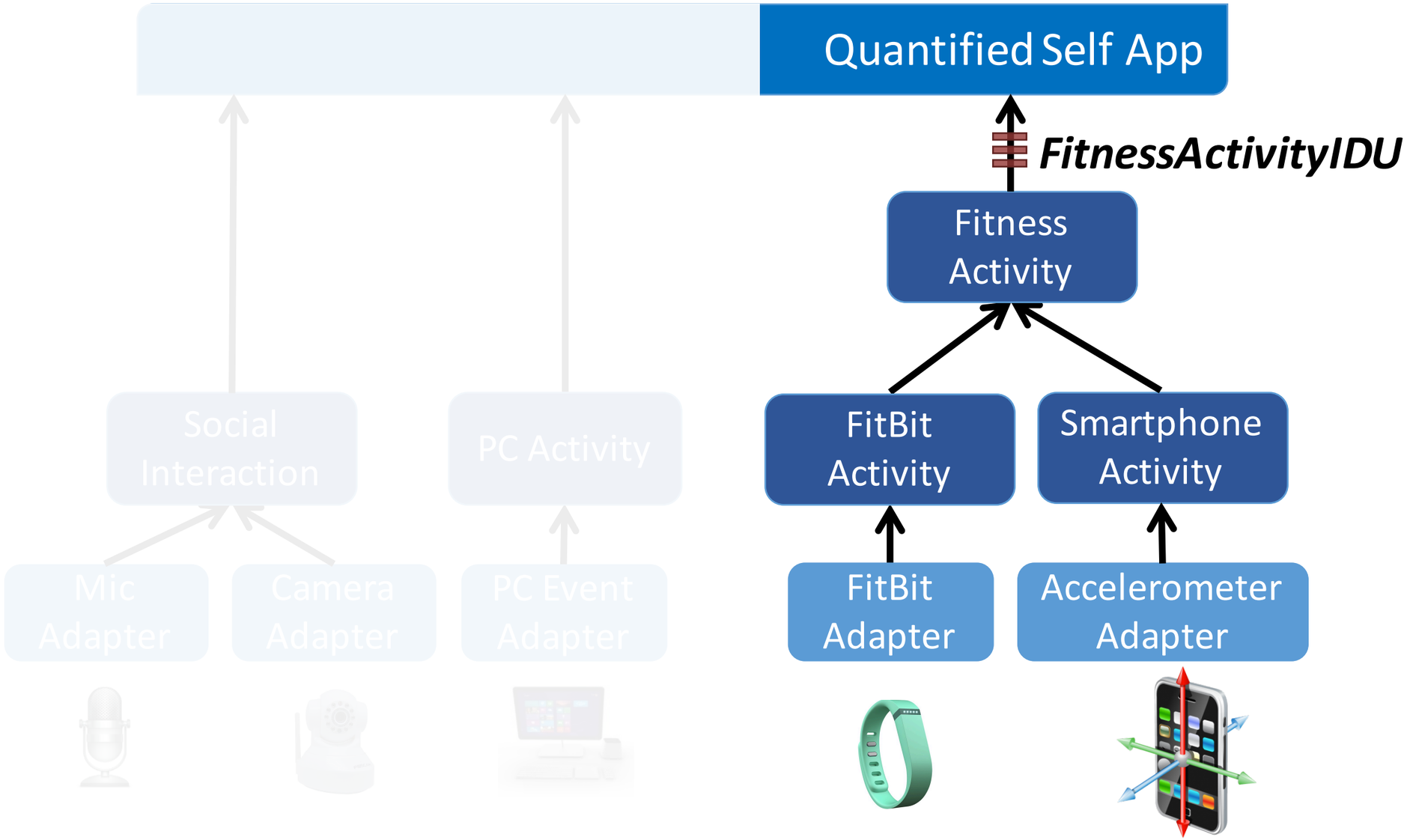
Outline

- Motivation and Beam overview
- Inference graph overview
- Key challenges addressed by the inference graph
- Evaluation of development effort

Quantified Self – Inference Graph

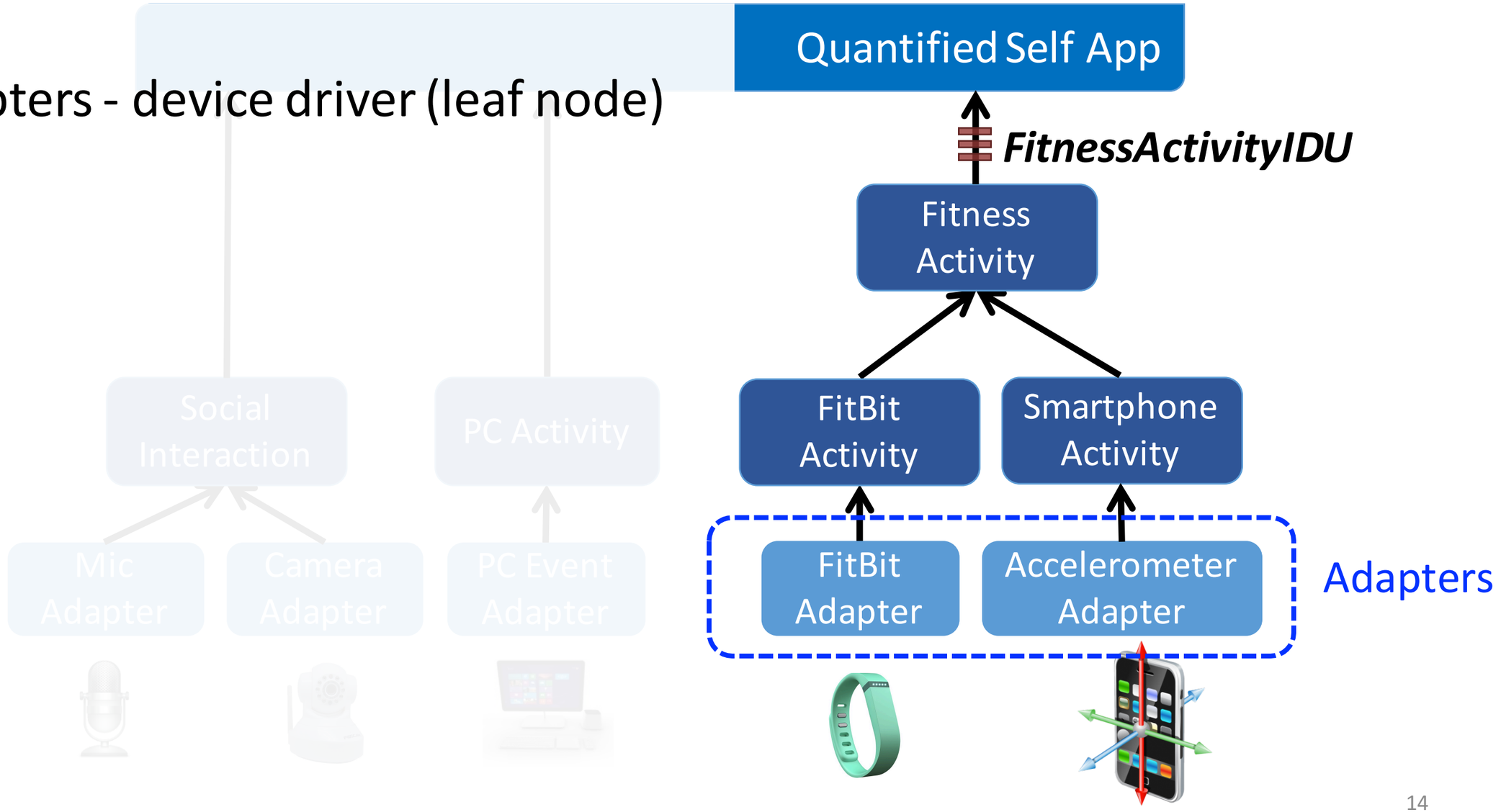


Inference Graph Components



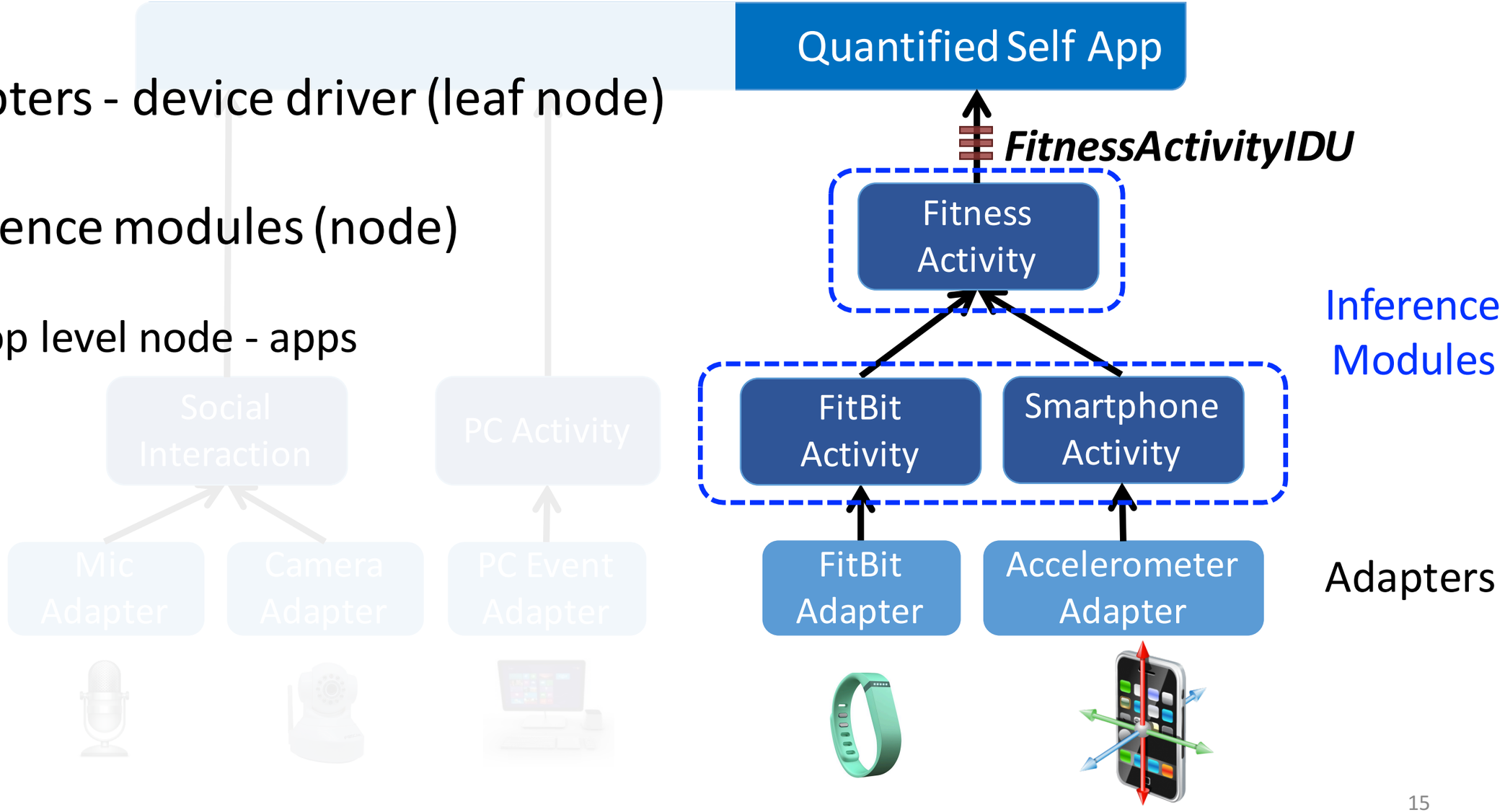
Inference Graph Components

- Adapters - device driver (leaf node)



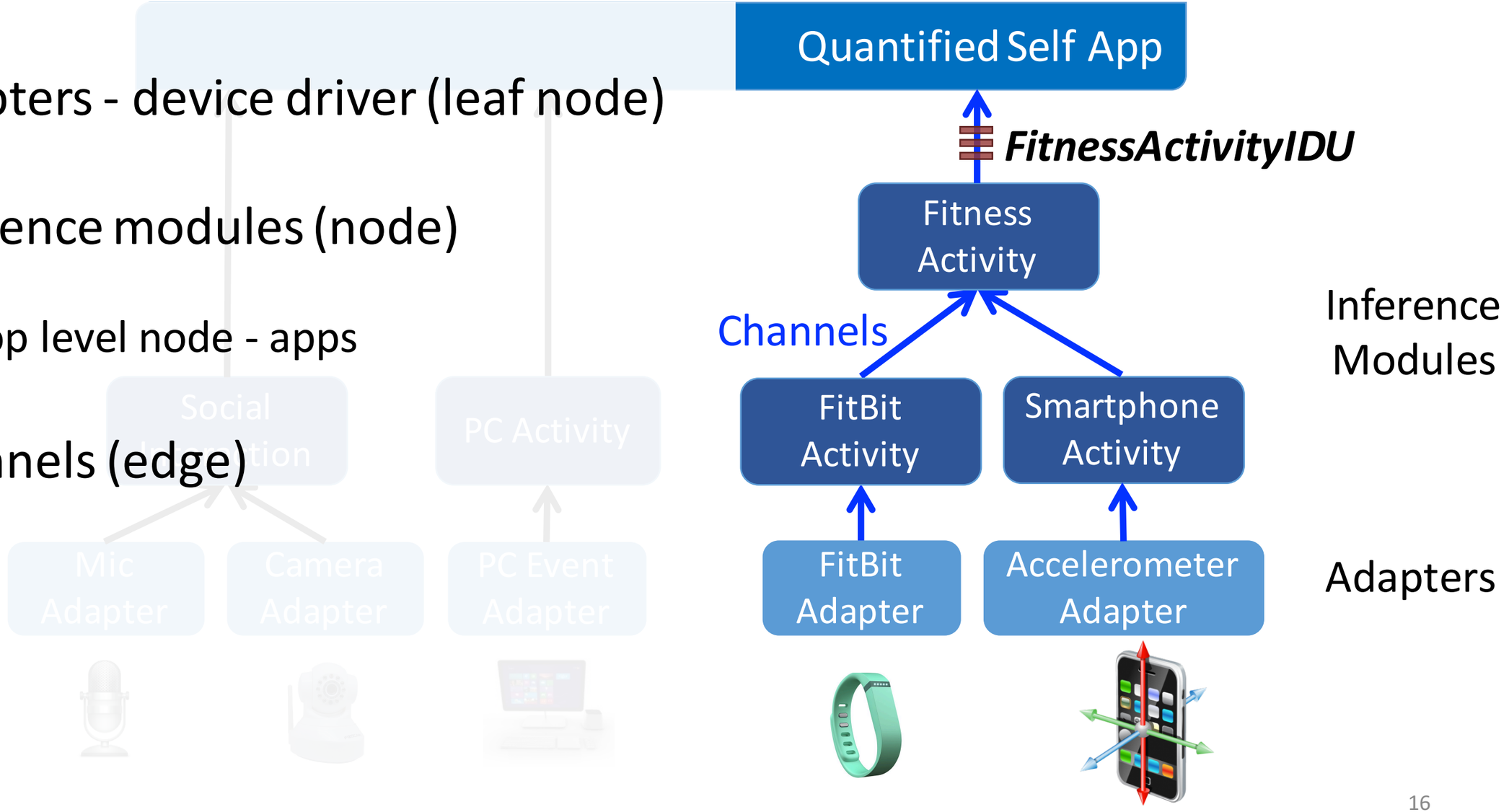
Inference Graph Components

- Adapters - device driver (leaf node)
- Inference modules (node)
  - Top level node - apps



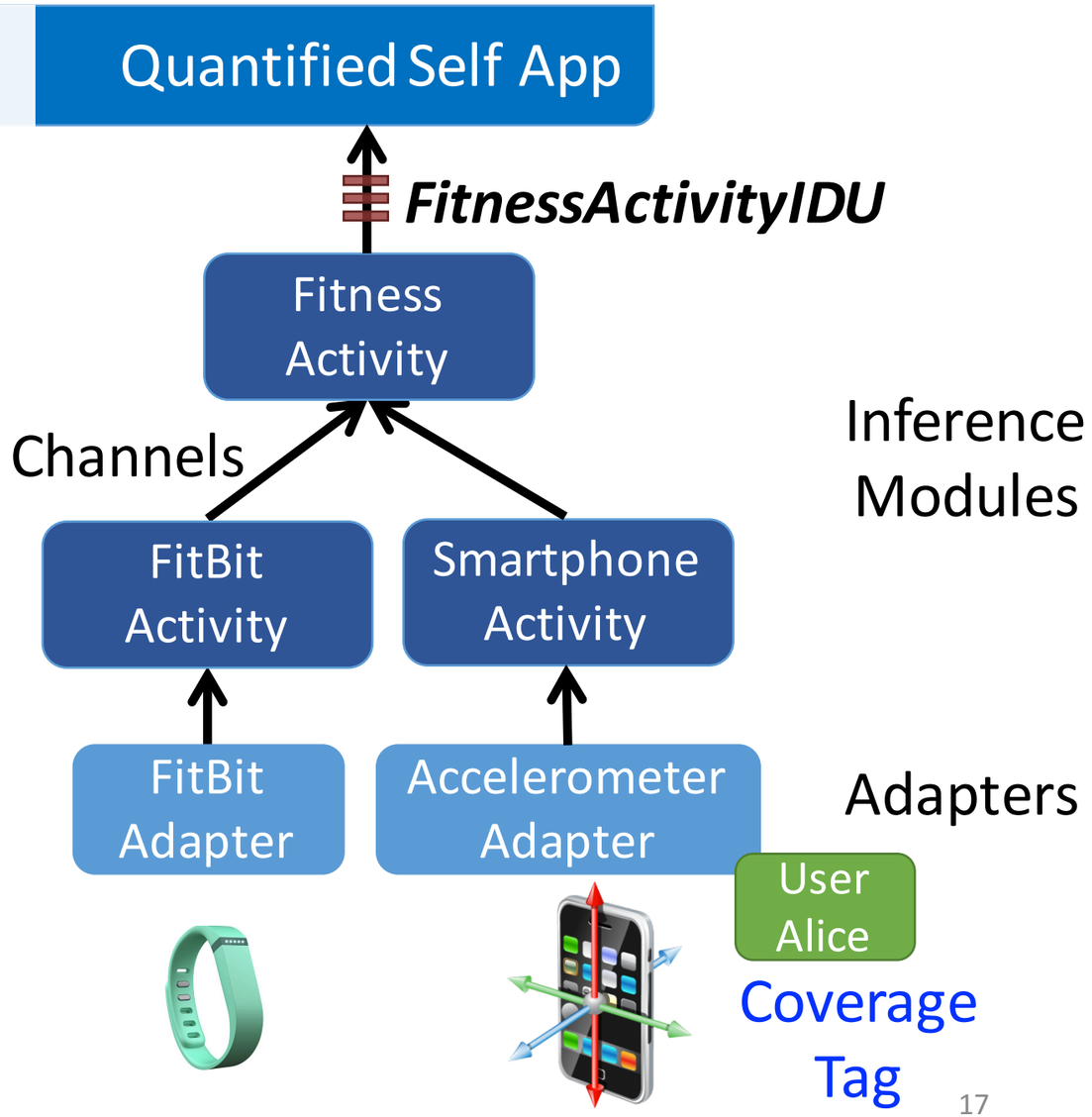
Inference Graph Components

- Adapters - device driver (leaf node)
- Inference modules (node)
  - Top level node - apps
- Channels (edge)

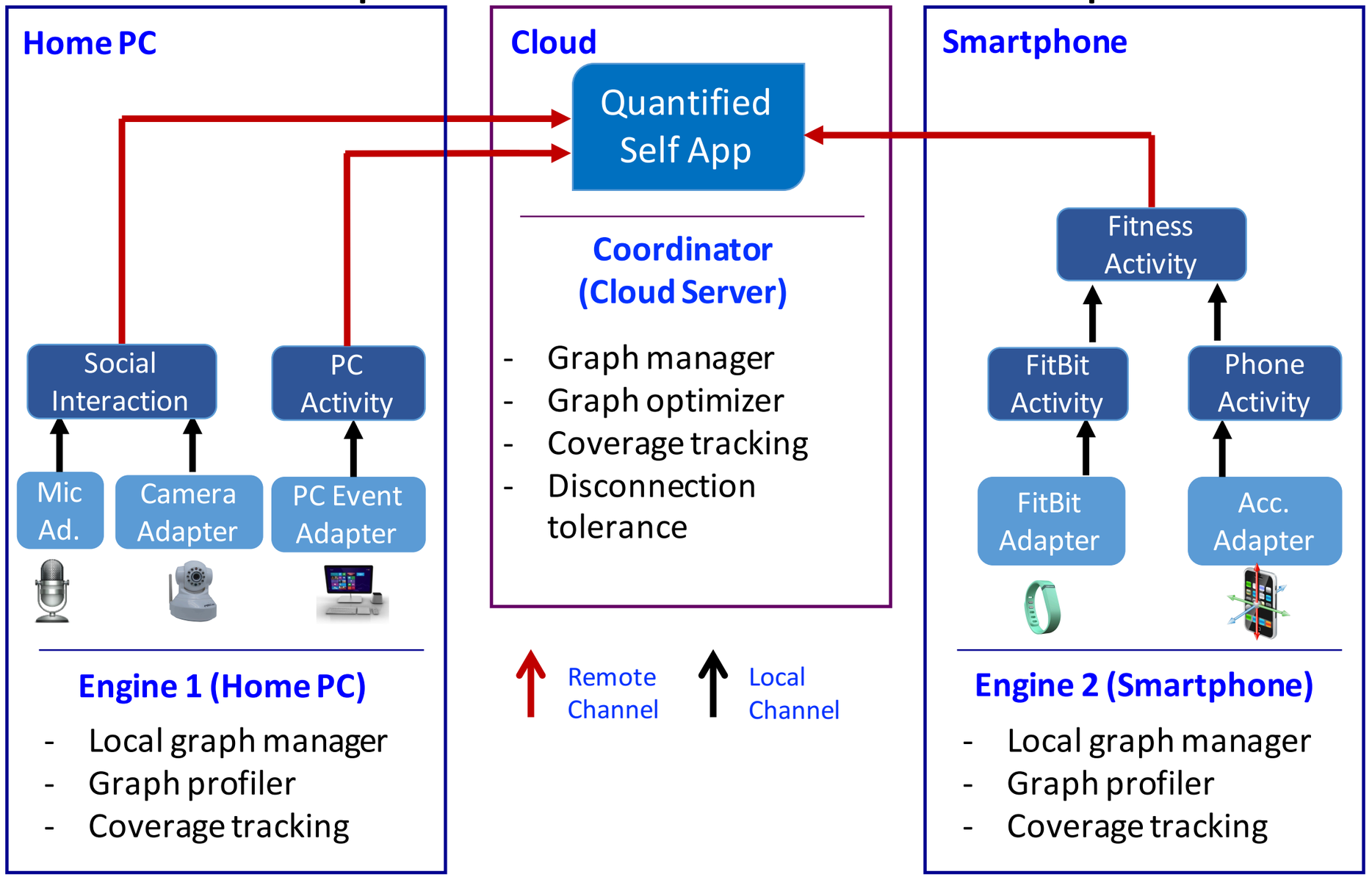


Inference Graph Components

- Adapters - device driver (leaf node)
- Inference modules (node)
  - Top level node - apps
- Channels (edge)
- Coverage tags
  - Manage sensor coverage



Inference Graph Runs Across Multiple Devices



Outline

- Motivation and Beam overview
- Inference graph overview
- Key challenges addressed by the inference graph
  - Device selection
  - Efficient resource usage
  - Disconnection tolerance (in our paper)
  - **Micro-benchmark results**
- Evaluation of development effort

Key Challenges Solved by the Inference Graph

Device Selection

- Select appropriate devices in a ***heterogeneous deployment*** that can satisfy an app's inference request
- Support settings with ***user mobility***

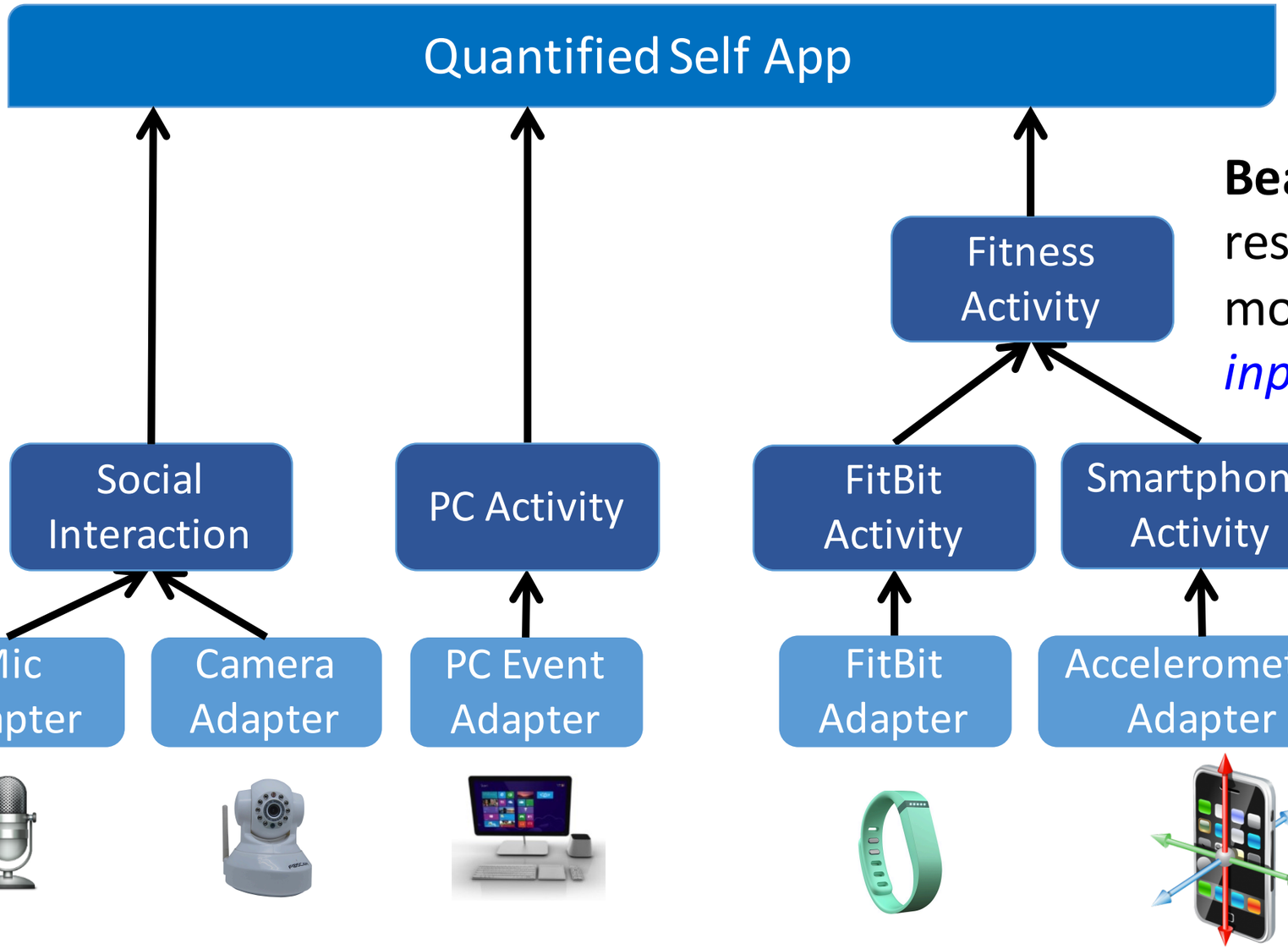
Efficient resource usage

- ***Efficiently partition*** computation across devices
- Optimize resource usage

Disconnection tolerance

- Handle dynamics caused by network ***disconnection*** and user mobility

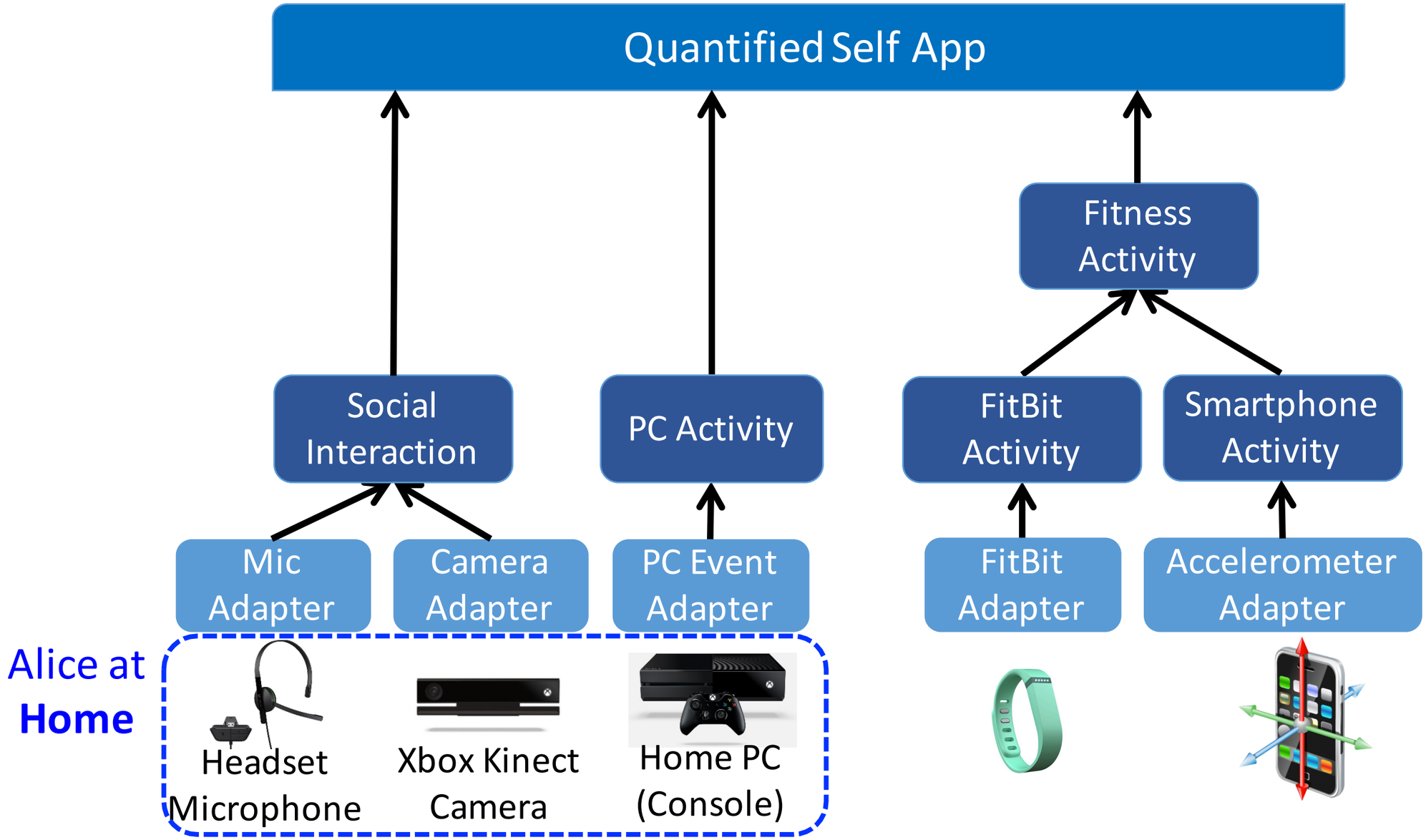
What Devices Should We Use?



Beam recursively resolves each module's *input dependency*.

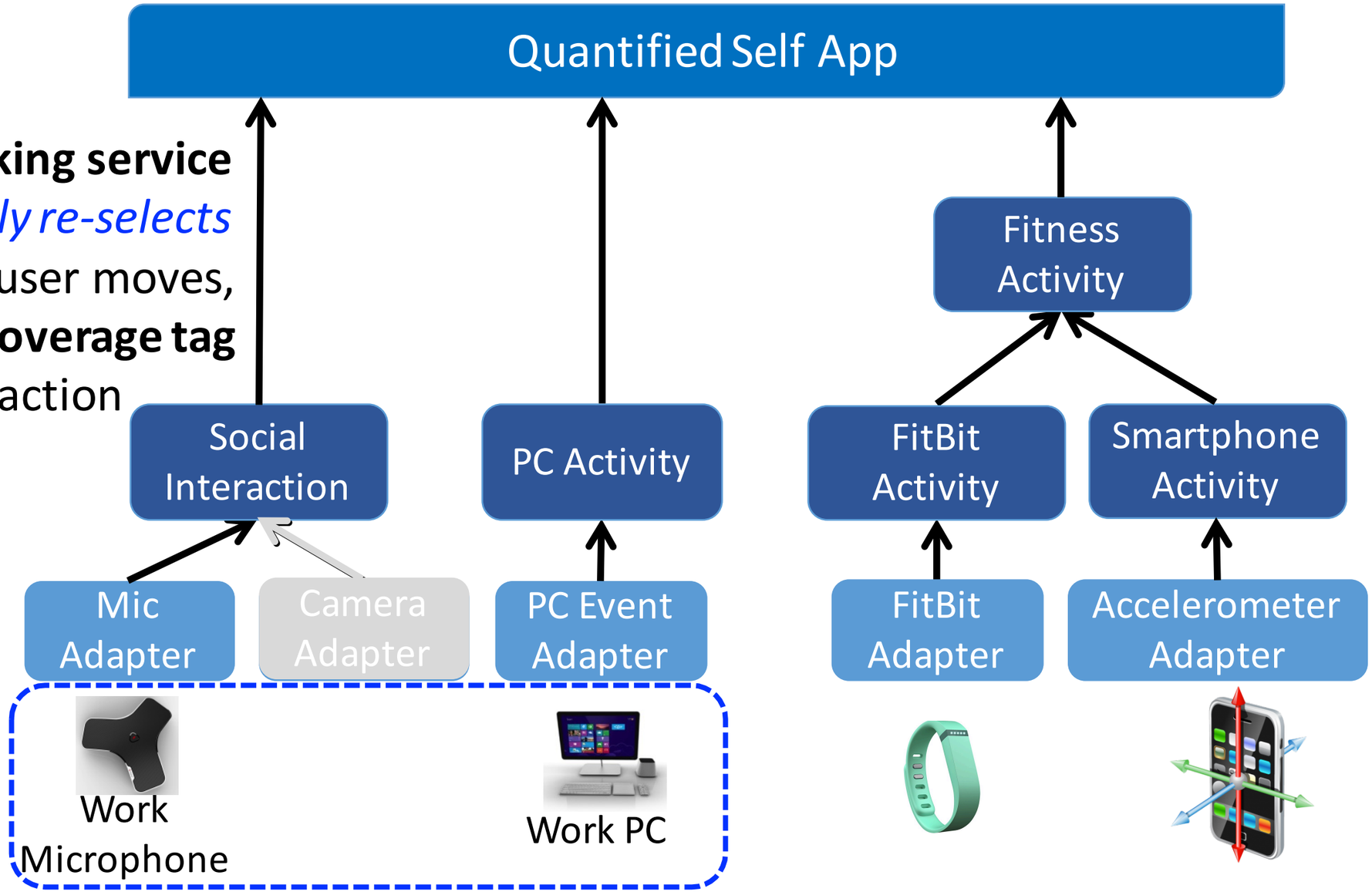
Heterogeneous Hardware Devices

What Devices Should We Use?



What Devices Should We Use?

Beam **tracking service** *dynamically re-selects devices* as user moves, using the **coverage tag** abstraction



Accuracy Improvement from Tracking

PC Activity Inference

0: Others

1: Mobile

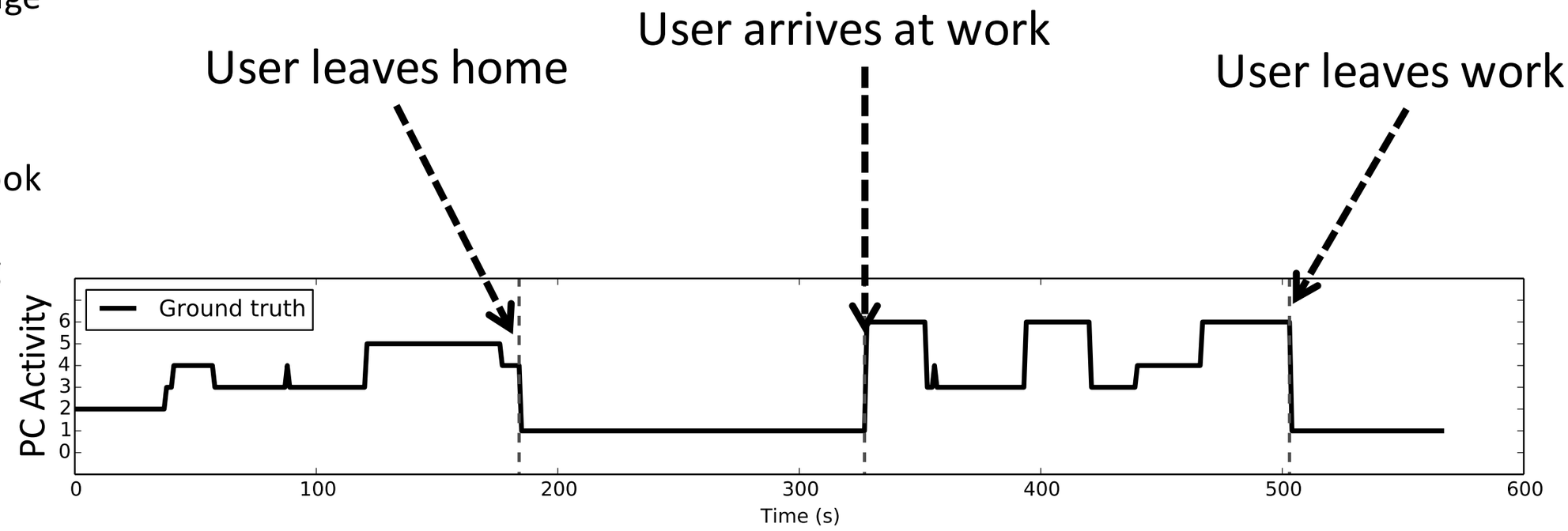
2: Reading

3: Webpage

4: Email

5: Facebook

6: Coding



Accuracy Improvement from Tracking

Inference Accuracy

29.68%

0: Others

1: Mobile

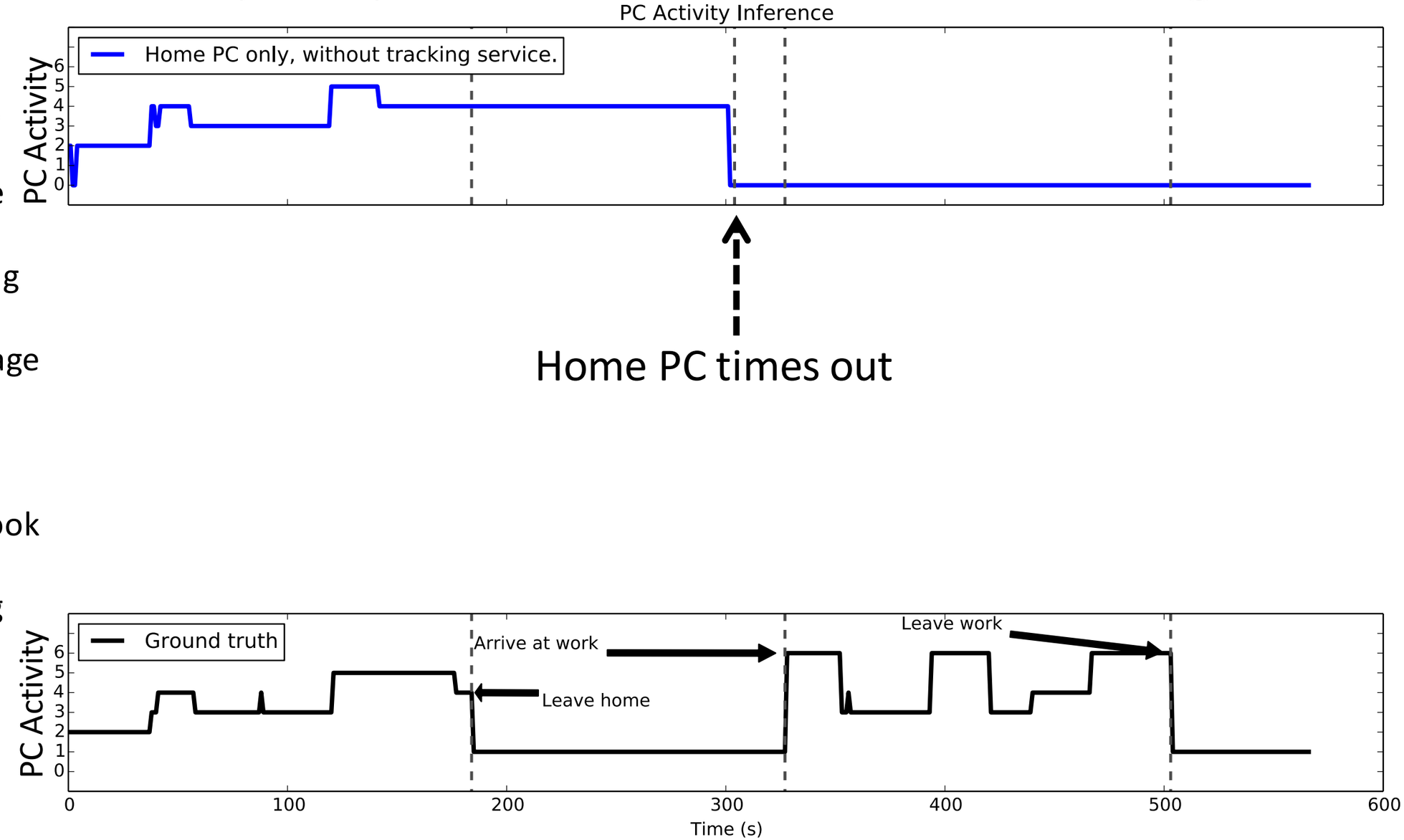
2: Reading

3: Webpage

4: Email

5: Facebook

6: Coding



Accuracy Improvement from Tracking

Inference Accuracy

0: Others

1: Mobile

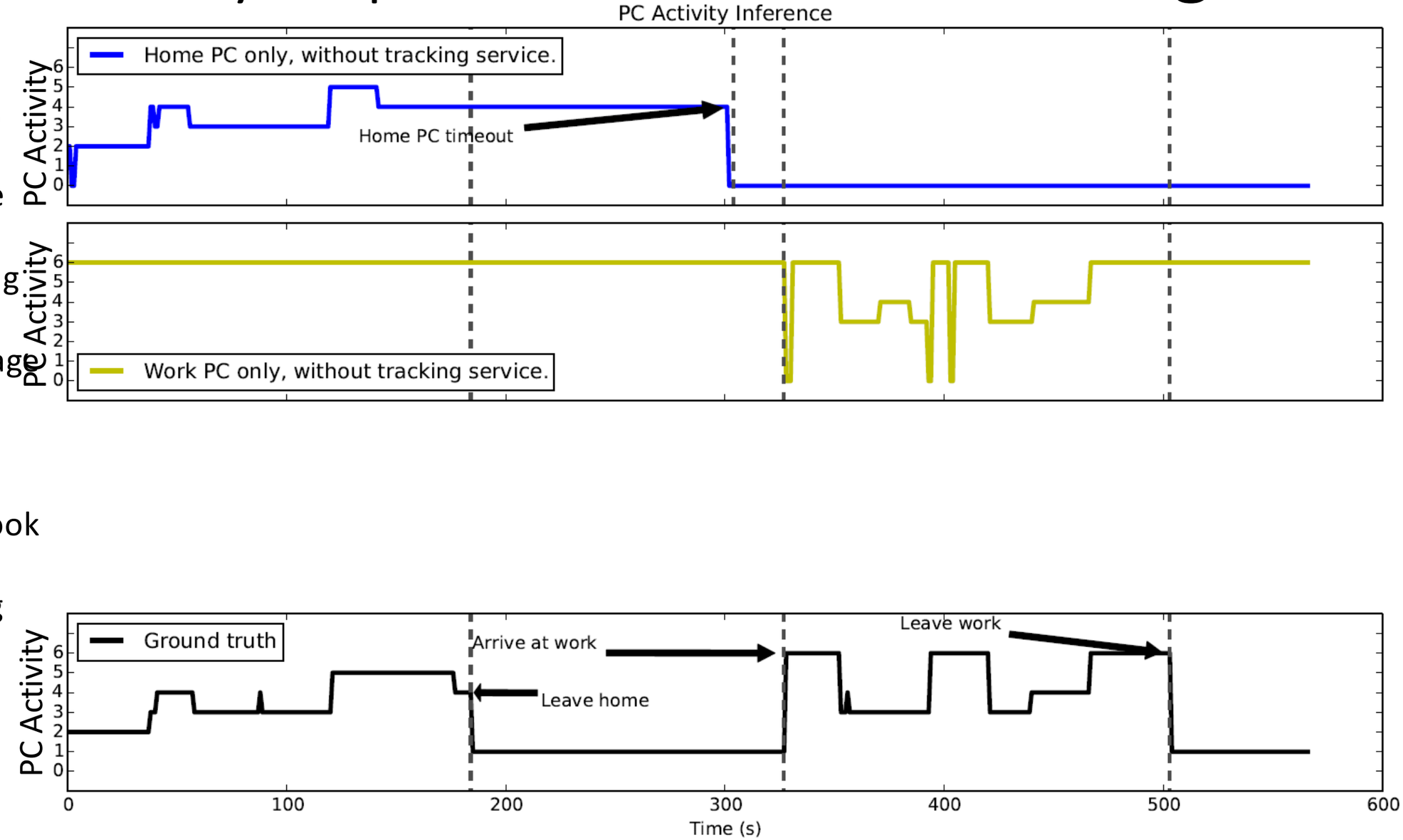
2: Reading

3: Webpage

4: Email

5: Facebook

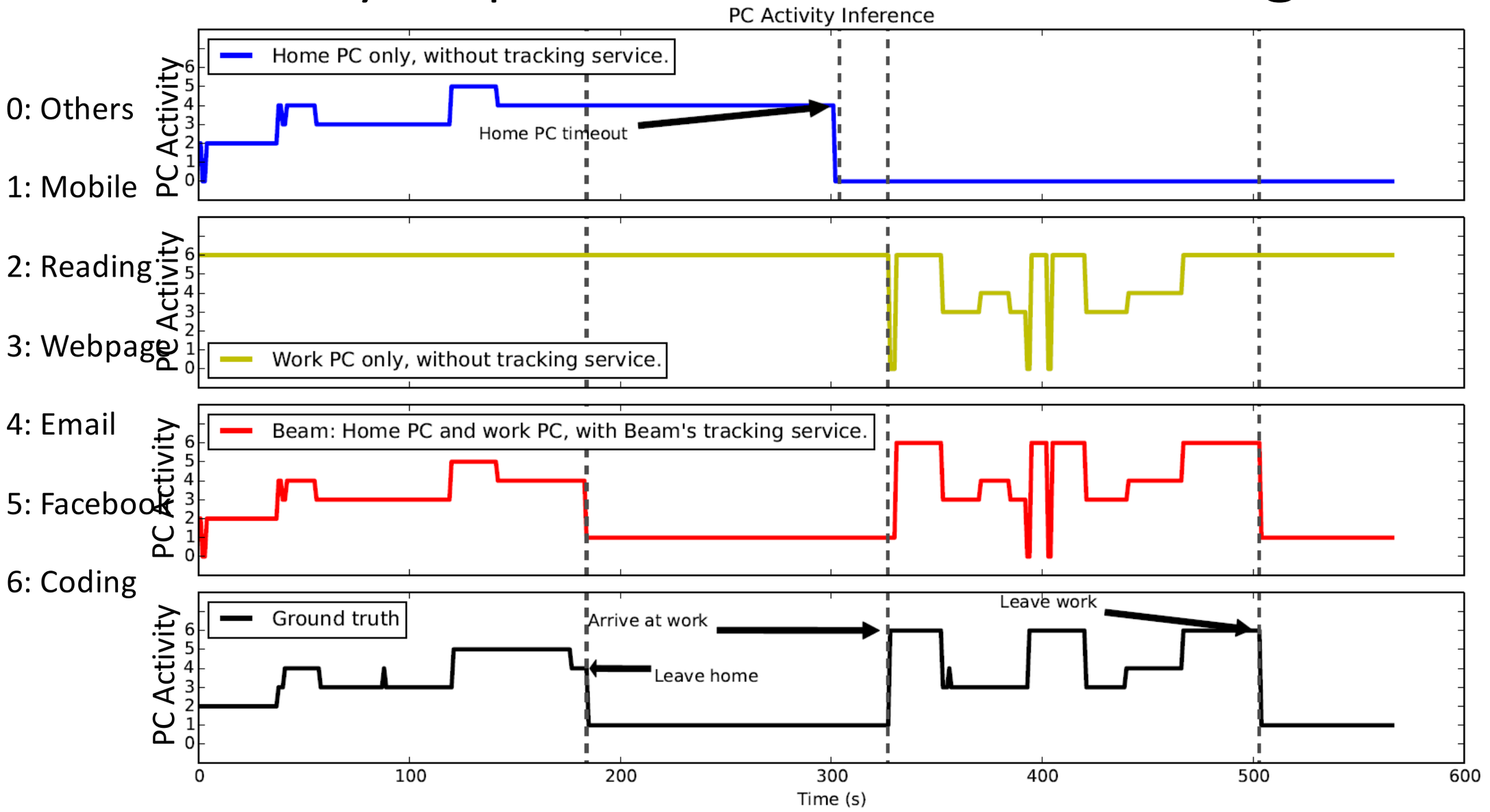
6: Coding



29.68%

26.94%

Accuracy Improvement from Tracking



Inference Accuracy

29.68%

26.94%

88.16%

~58% better

Key Challenges Solved by the Inference Graph

Device Selection

- Select appropriate devices in a ***heterogeneous deployment*** that can satisfy an app's inference request
- Support settings with ***user mobility***

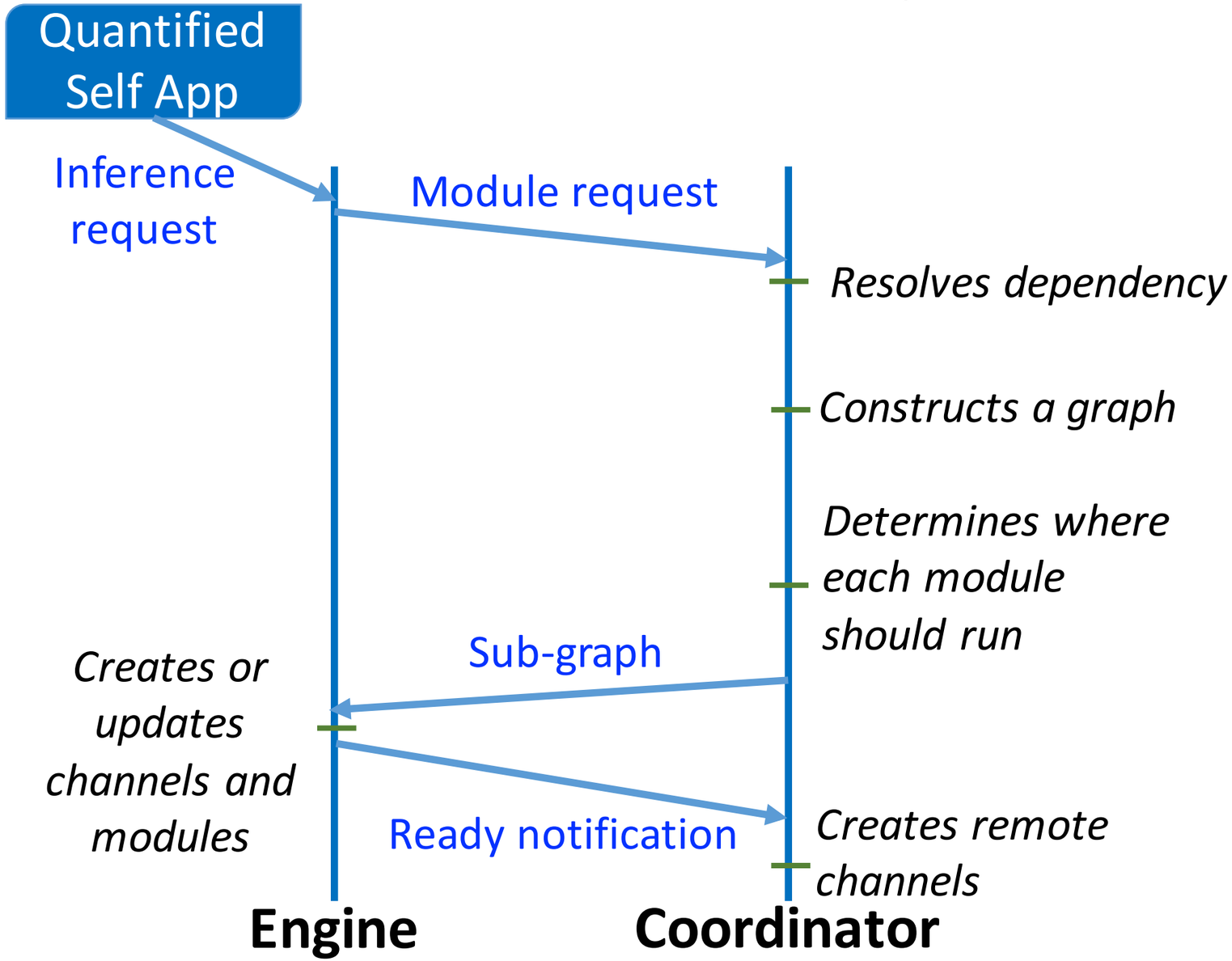
Efficient resource usage

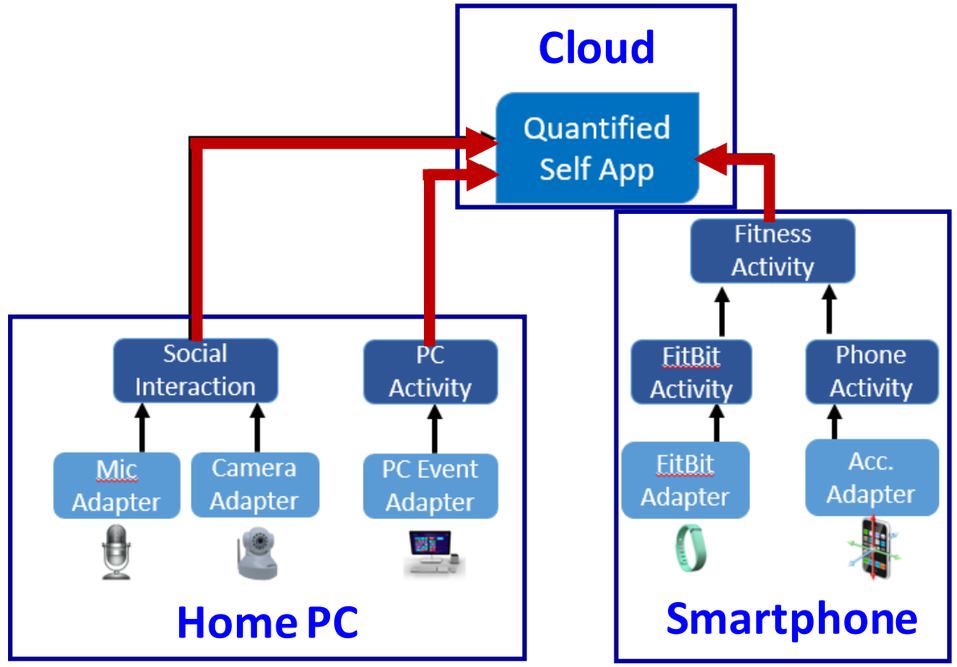
- ***Efficiently partition*** computation across devices
- Optimize resource usage

Disconnection tolerance

- Handle dynamics caused by network ***disconnection*** and user mobility

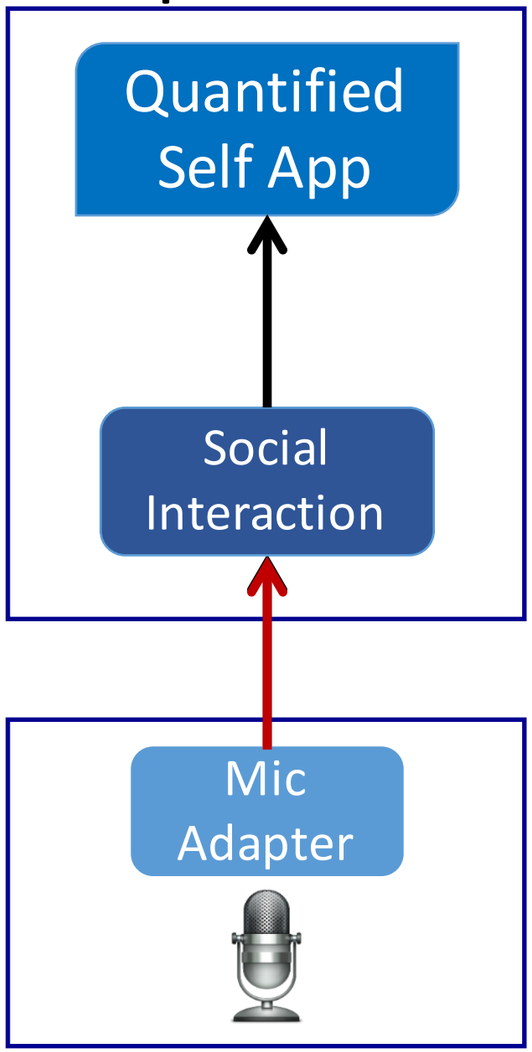
Where Should Computation Run?





Beam Optimization - Reactive

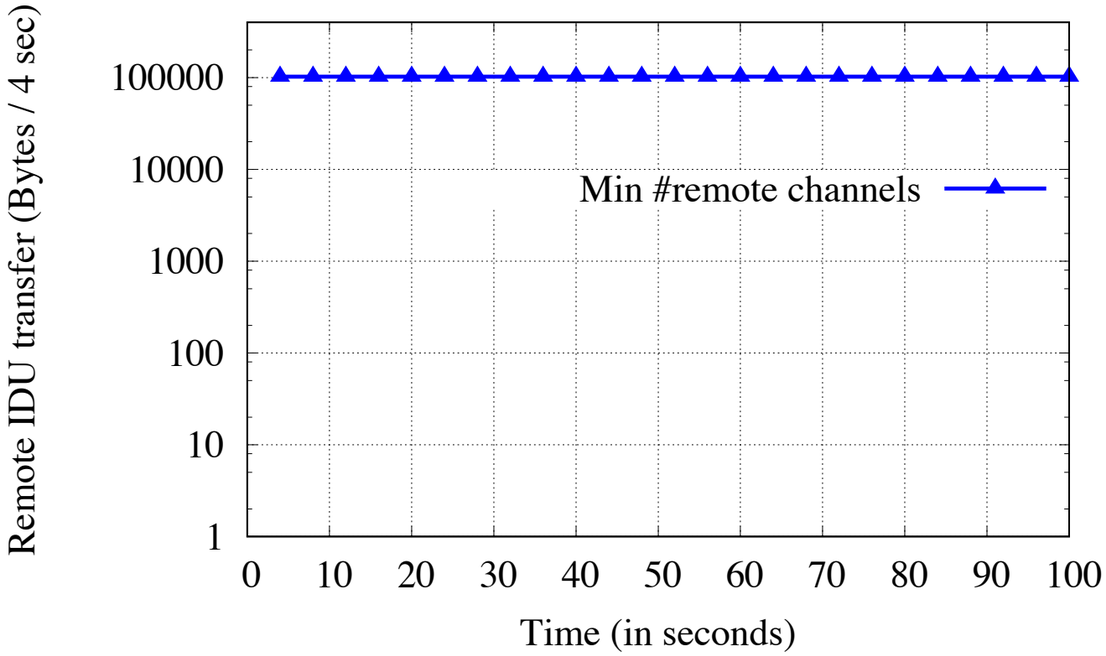
Cloud



Home PC

Reactive:
Minimize # of remote channels

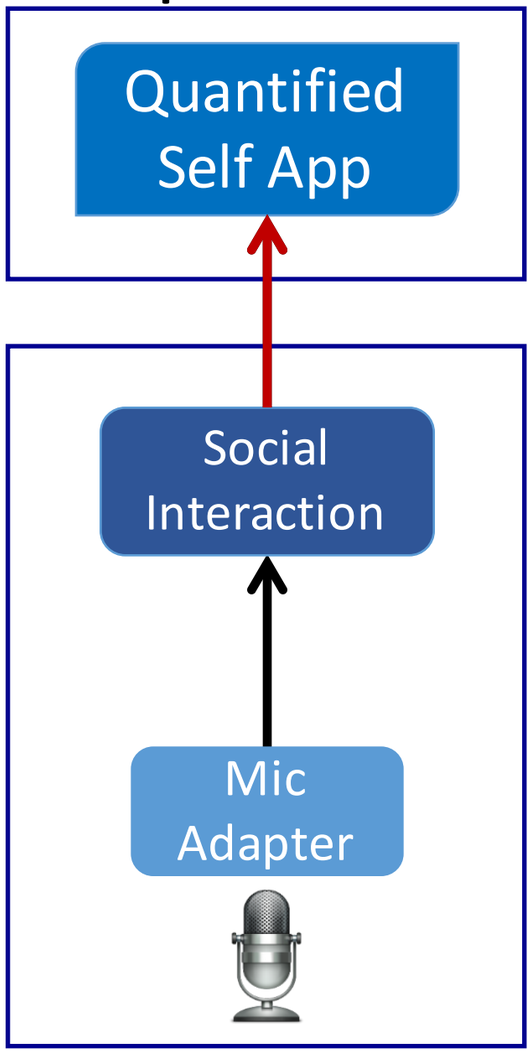
Wide-area data transfer, 100 second



Beam's ***reactive optimization*** minimizes # of remote channels, but results in high remote data transfer rate

Beam Optimization - Proactive

Cloud

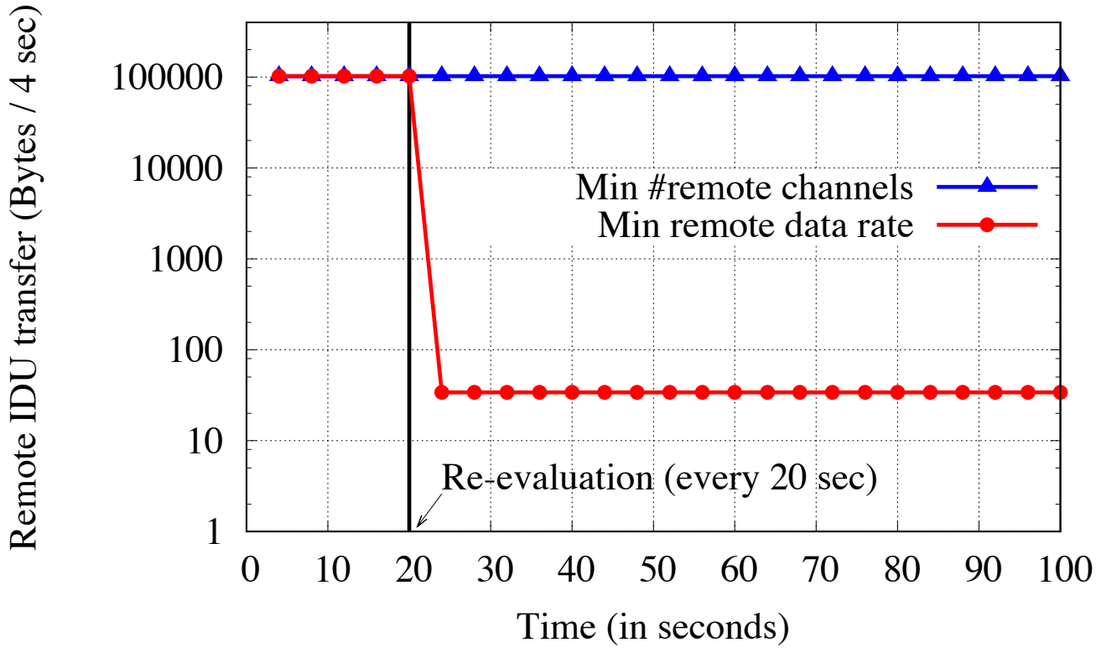


Home PC

Proactive:

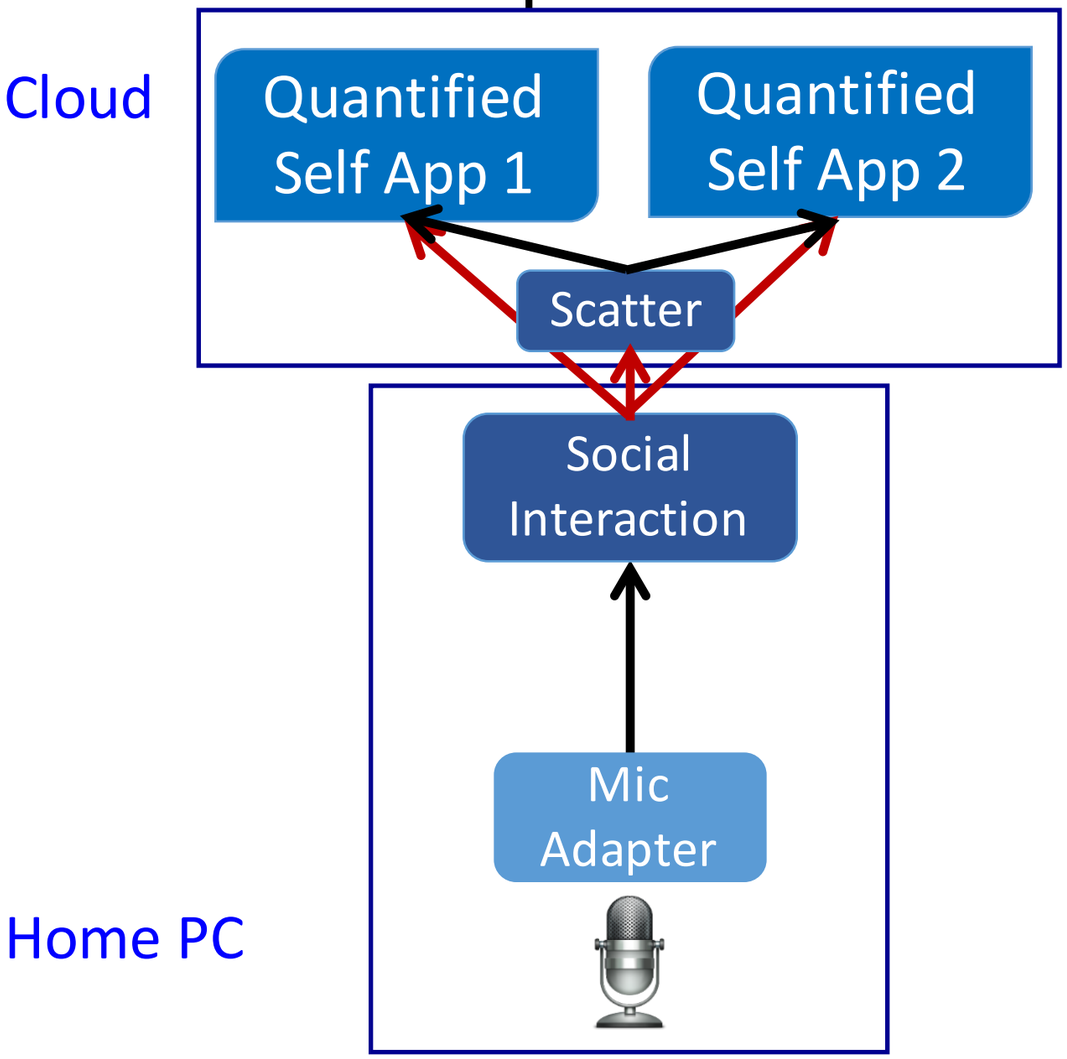
Active profiling, minimize remote data rate

Wide-area data transfer, 100 second



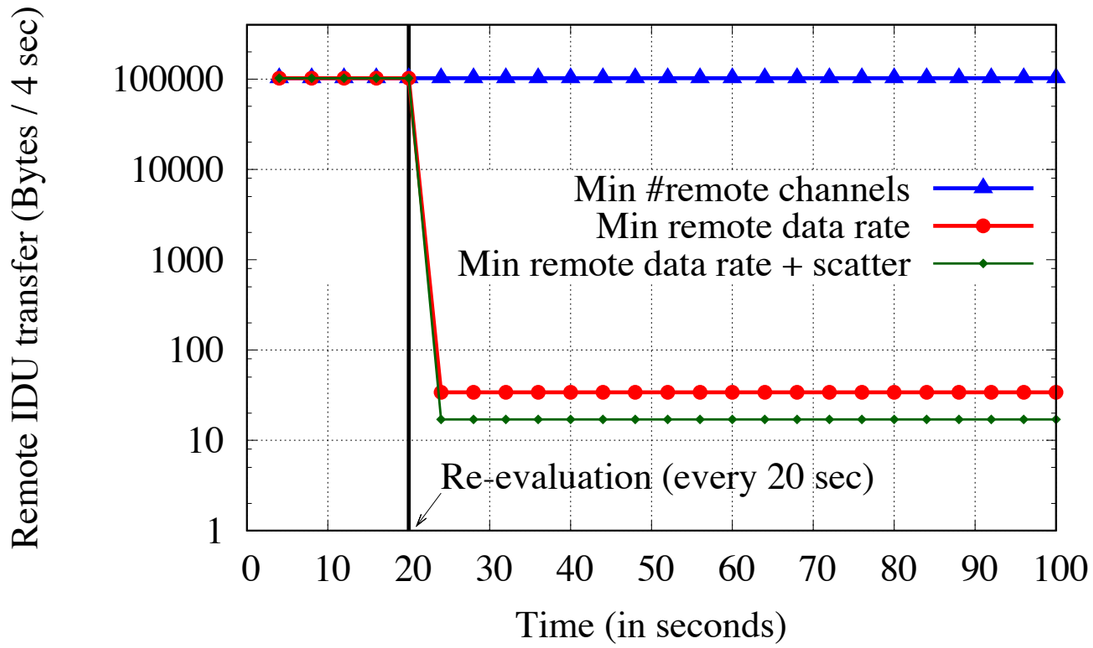
Beam's ***proactive optimization*** identifies high remote data transfer rate and re-evaluate graph

Beam Optimization – Scatter Node



Scatter node optimization

Wide-area data transfer, 100 second



Beam's ***scatter node optimization*** further reduces remote data transfer

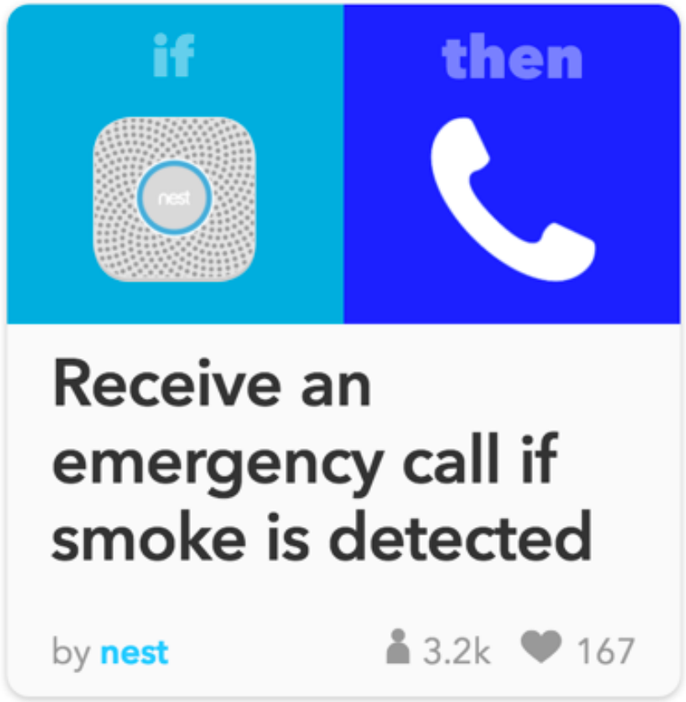
Beam Implementation

- C# cross-platform portable service
  - Supports .NET v4.5, Windows Store 8.1, and Windows Phone 8.1 apps
- Sample implementation of 8 inference modules and 9 adapters
  - Including a HomeOS adapter for more device abstractions
- 9609 total source lines of code
- APIs for both app developers and inference developers

Outline

- Motivation and Beam overview
- Inference graph overview
- Key challenges addressed by the inference graph
- Evaluation of development effort

Sample Apps: Quantified Self and IFTTT Rules

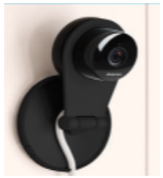


Quantified Self App

IFTTT Rules App







Evaluation of Development Effort

Monolithic-All Cloud (M-AC)

Monolithic-Cloud and Device (M-CD)

Monolithic-Inference Library (M-Lib)

Monolithic-Sensor Hub (M-Hub)

Beam

Evaluation of Development Effort

Monolithic-All Cloud (M-AC)

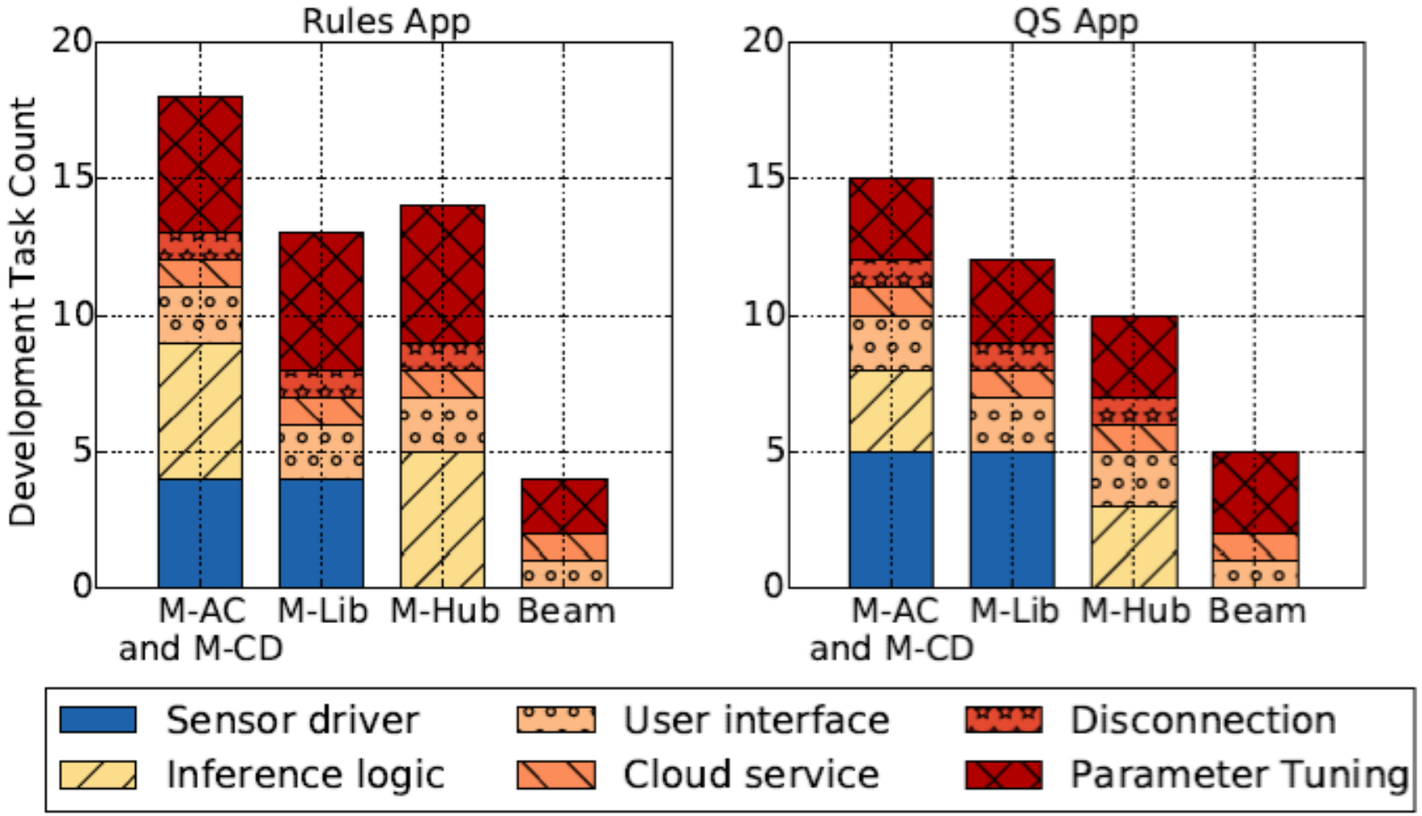
Monolithic-Cloud and Device (M-CD)

Monolithic-Inference Library (M-Lib)

Monolithic-Sensor Hub (M-Hub)

Beam

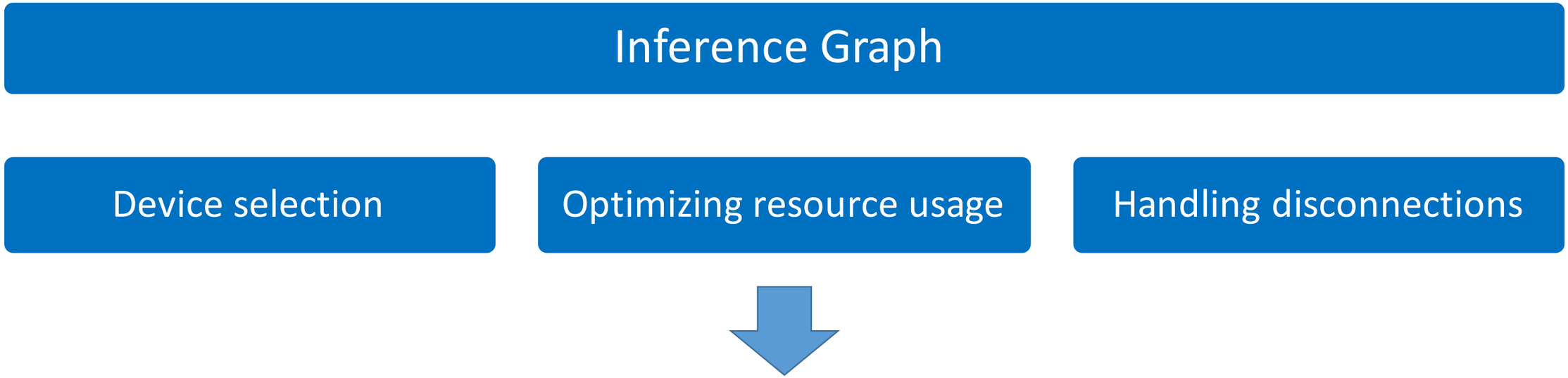
Number of *development tasks*



Up to 4.5x lower number of dev tasks, and up to 12x lower source lines of code

Conclusion

Decouple "what is sensed and inferred" from "how it is sensed and inferred"



- Up to *4.5x lower number of tasks* and *12x lower source line of code* in application development effort
- Up to *3x higher inference accuracy* from dynamic device selection
- Beam's *dynamic optimizations* match hand-optimized apps