Differential Energy Profiling: Energy Optimization via Diffing Similar apps

Abhilash Jindal

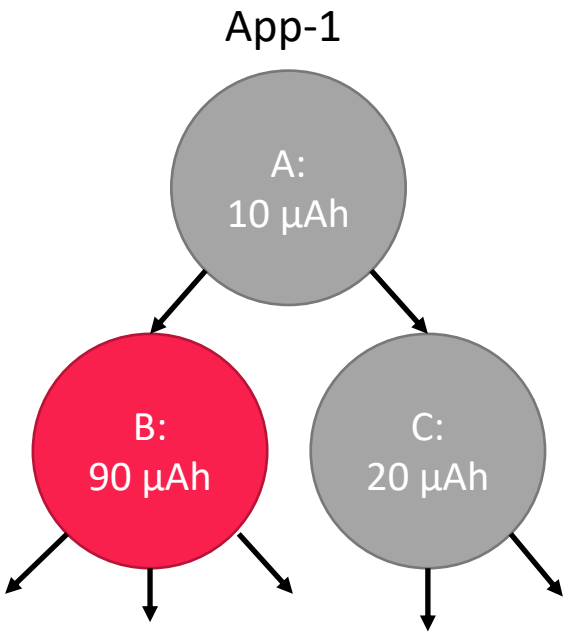
Y. Charlie Hu

PURDUE UNIVERSITY

mobile enerlytics

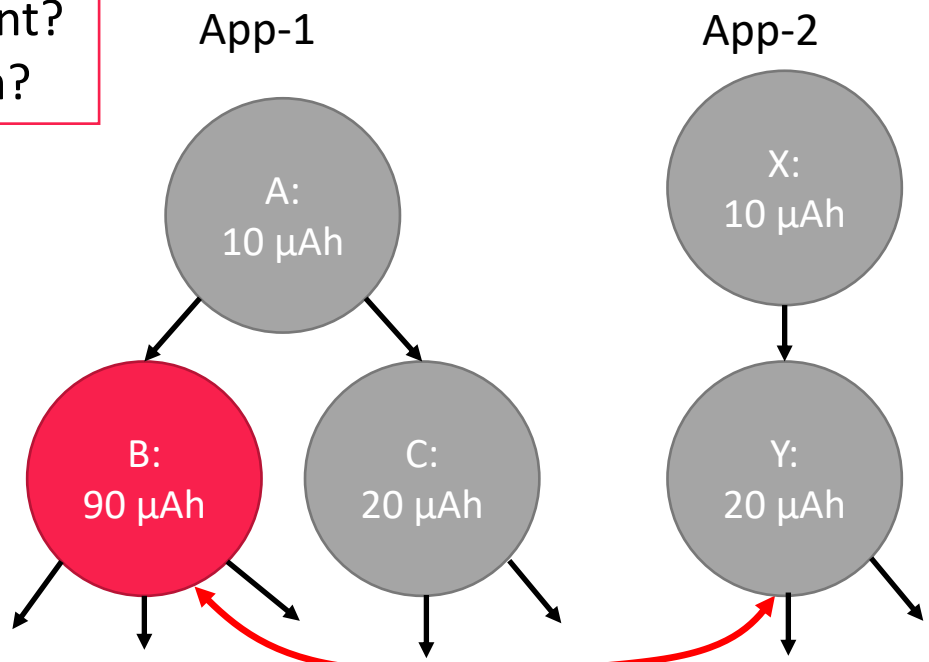
Towards optimizing app battery drain

Source code energy profilers



Is there room for improvement?
How to do the optimization?

DIFFPROF (this paper)

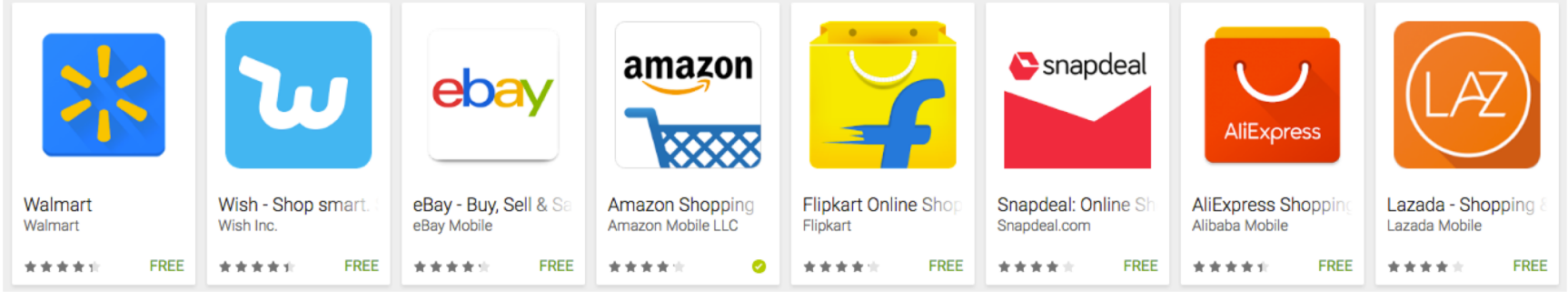


Part I: Why diffing?

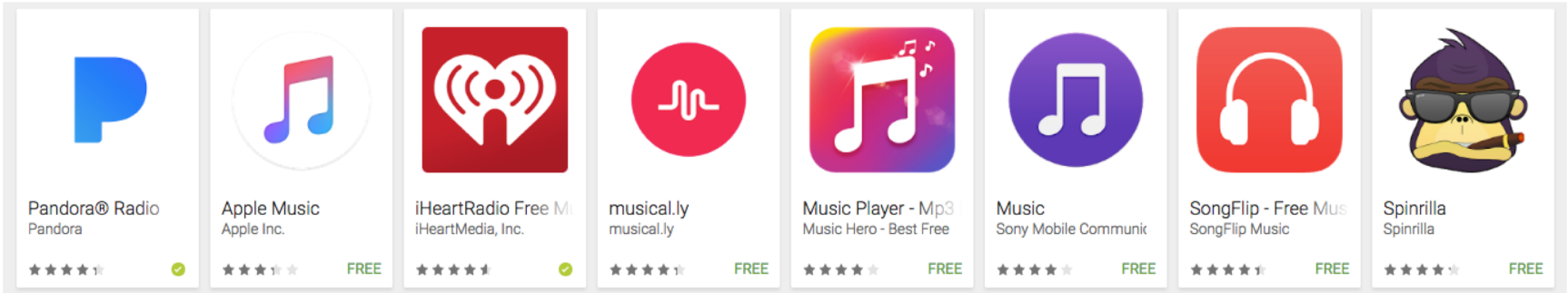
Key Observations

There is an app for that!

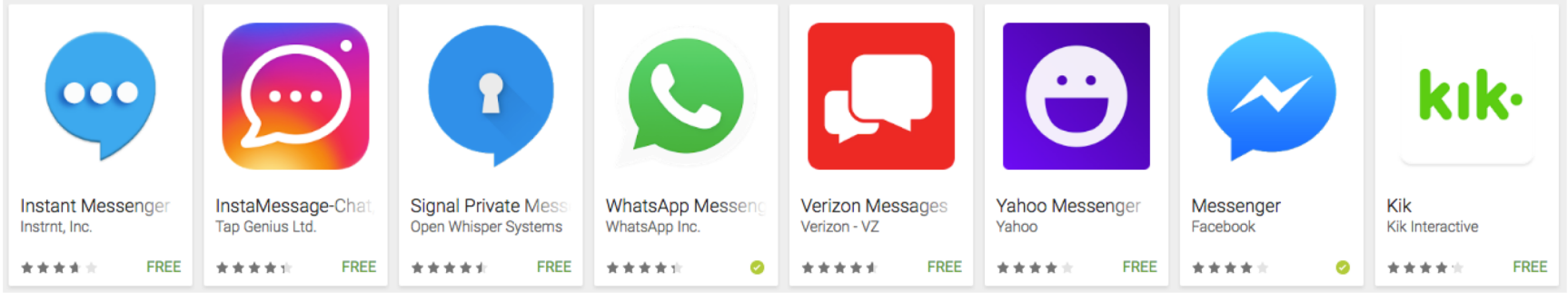
Online Shopping



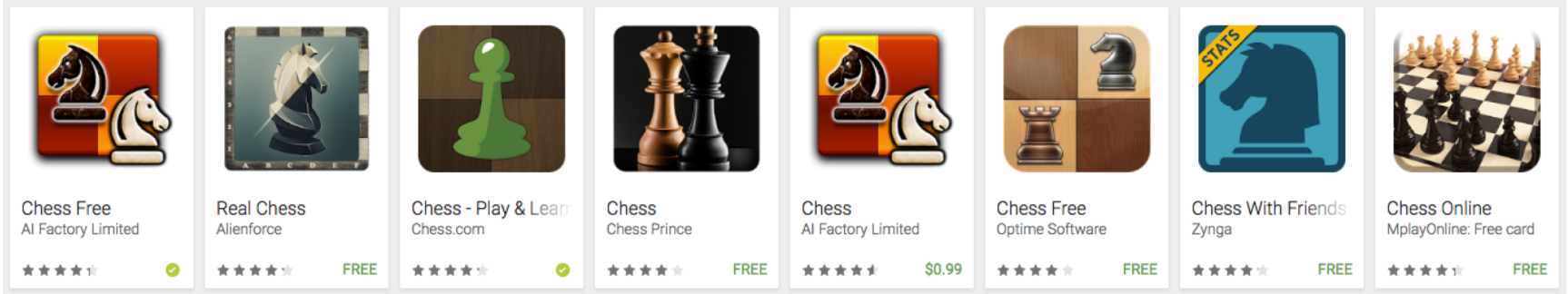
Music



Instant Message

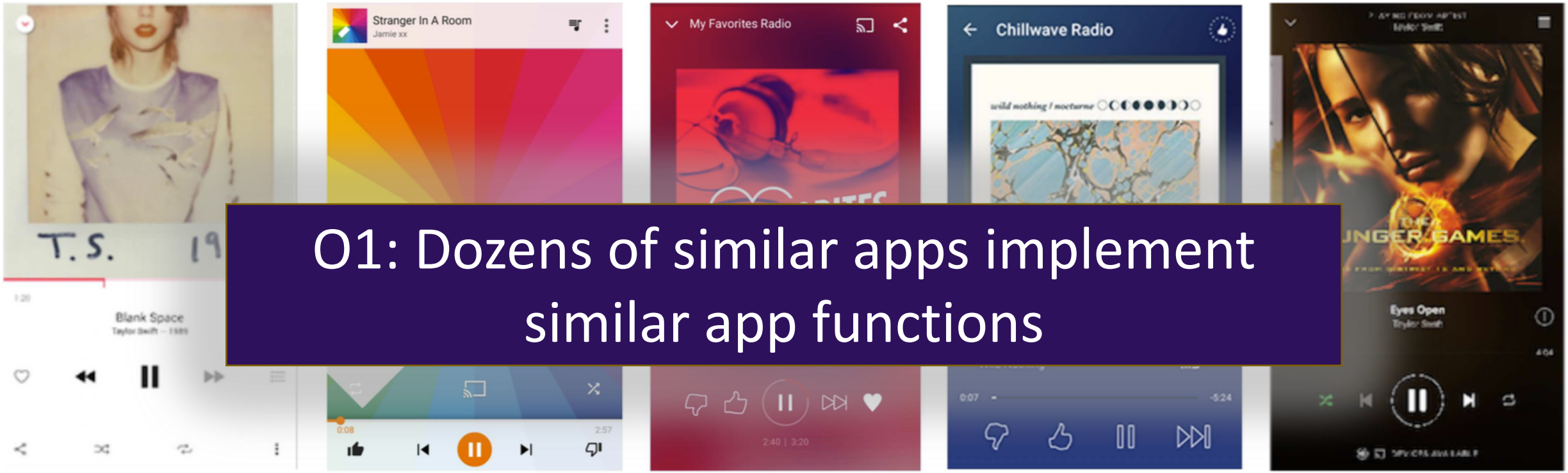


Chess



There are dozens of apps for that!

Similar apps implement similar functionalities

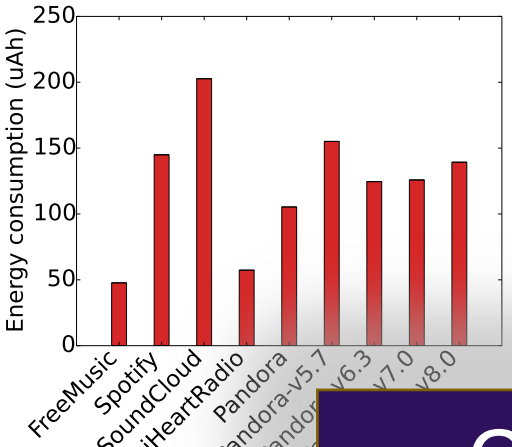


O1: Dozens of similar apps implement similar app functions

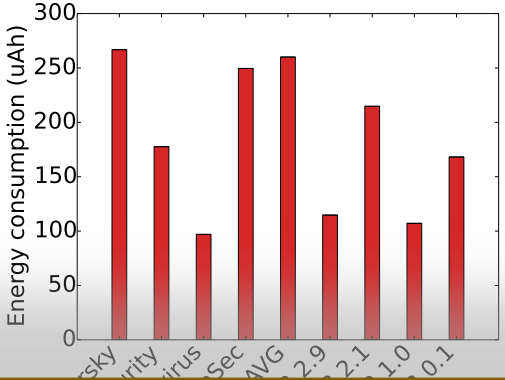
(a) Apple Music (b) Google Play music (c) iHeartRadio (d) Pandora (e) Spotify

Battery drain of similar apps differ a lot

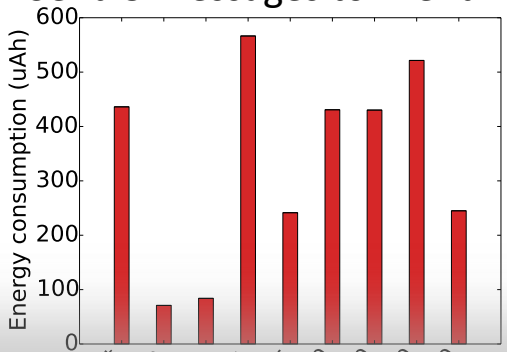
Music playback for 30 sec



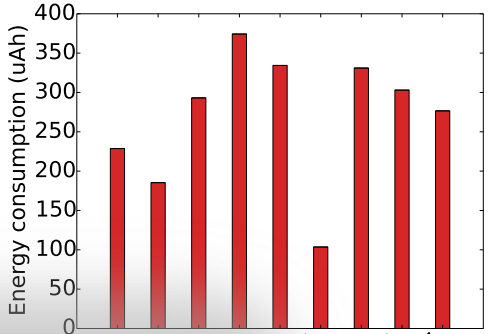
Press scan and wait



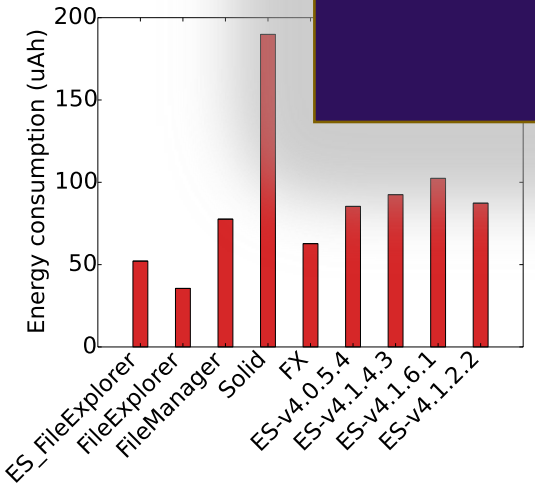
Send 5 messages to friend



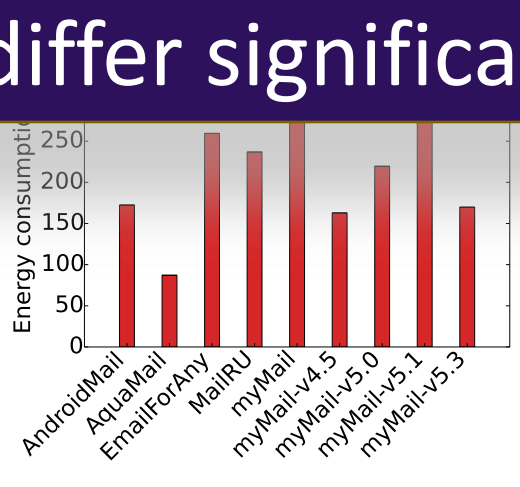
Press clean and wait



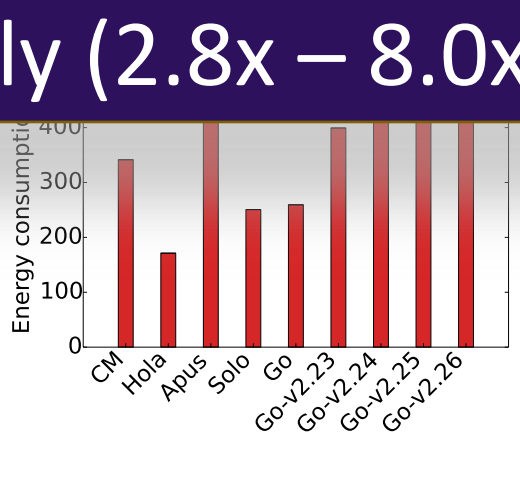
O2: Battery drain among similar apps differ significantly (2.8x – 8.0x)



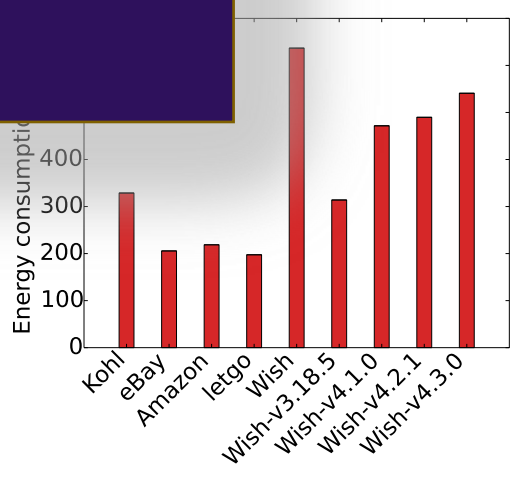
Create new folder, scroll



Compose email and send



Swipe 2 sec for 1 min

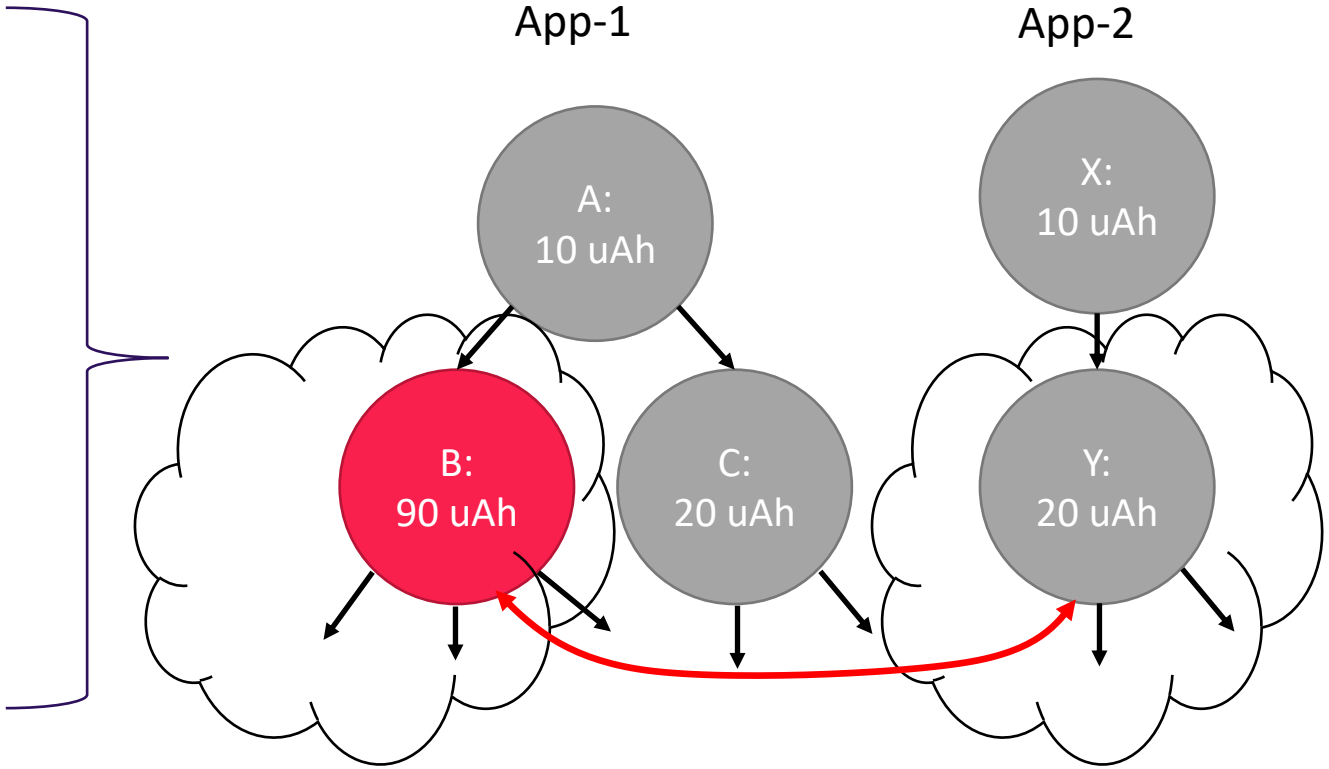


Search “socks”, scroll

Comparing energy profiles can potentially be effective

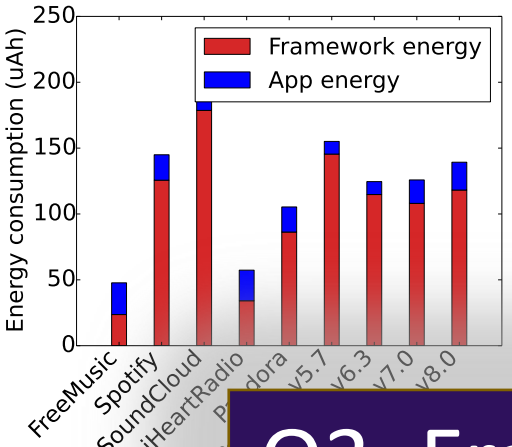
O1: Dozens of similar apps

O2: Battery drain differ significantly

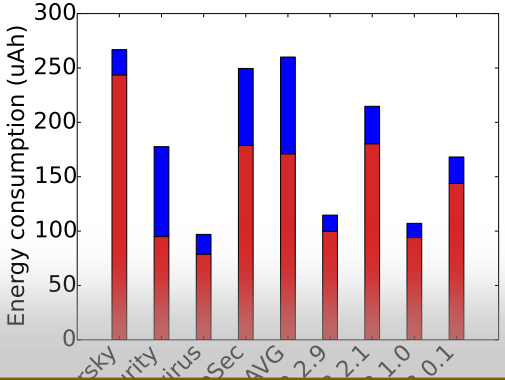


Framework services dominate energy drain

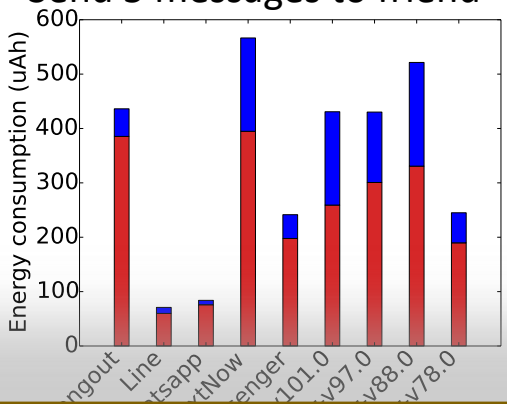
Music playback for 30 sec



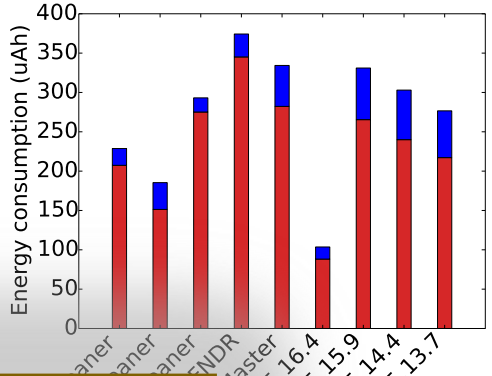
Press scan and wait



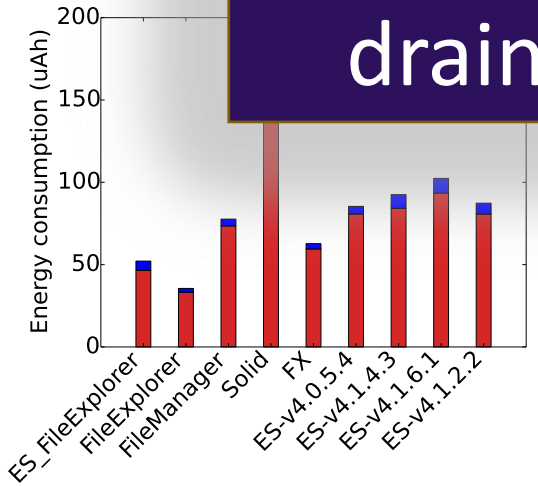
Send 5 messages to friend



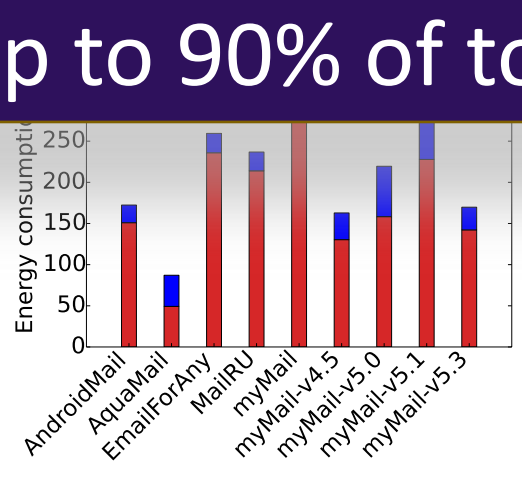
Press clean and wait



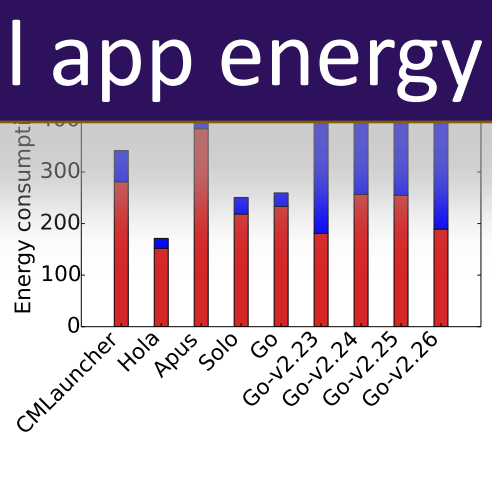
O3: Framework services, common to all apps, drain up to 90% of total app energy drain



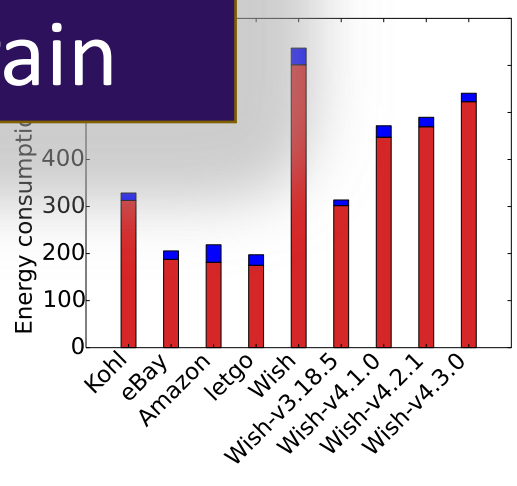
Create new folder, scroll



Compose email and send



Swipe 2 sec for 1 min



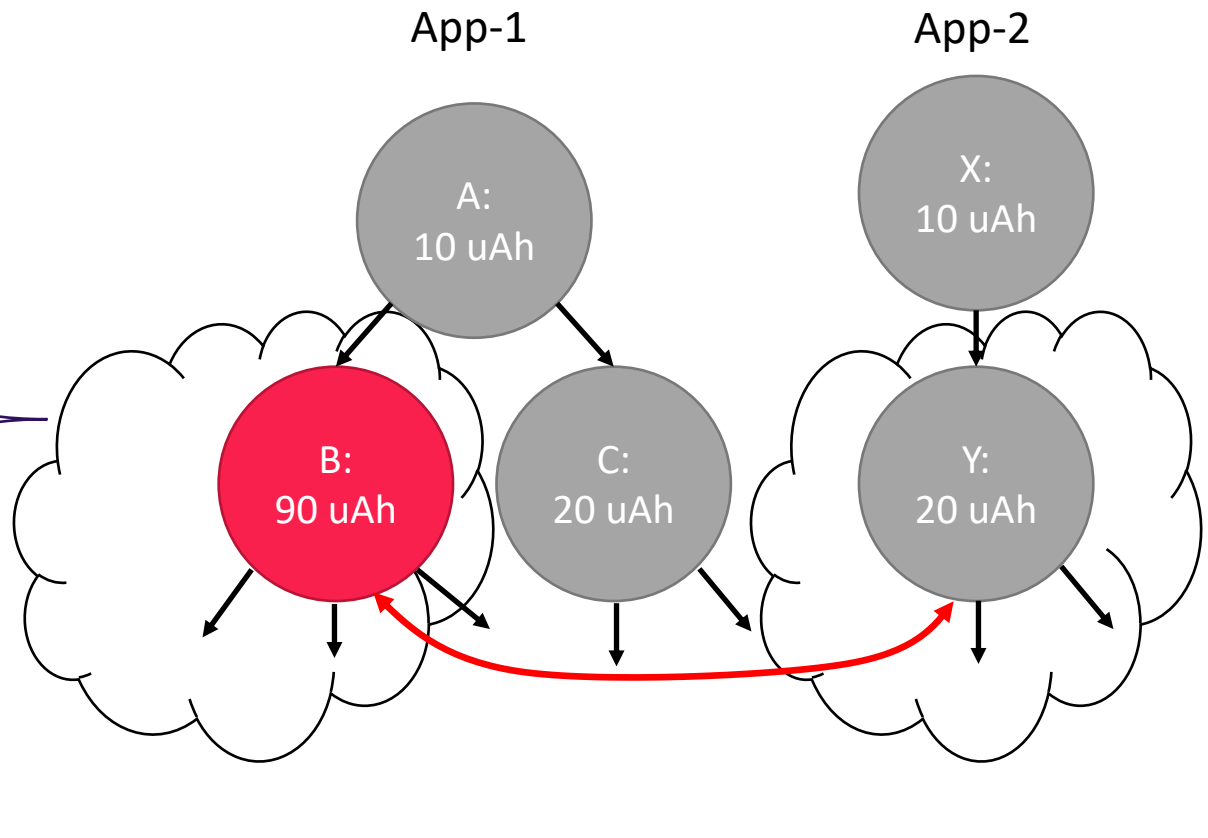
Search “socks”, scroll

Comparing energy profiles will be effective

O1: Dozens of similar apps

O2: Battery drain differ significantly

O3: Framework services dominate battery drain



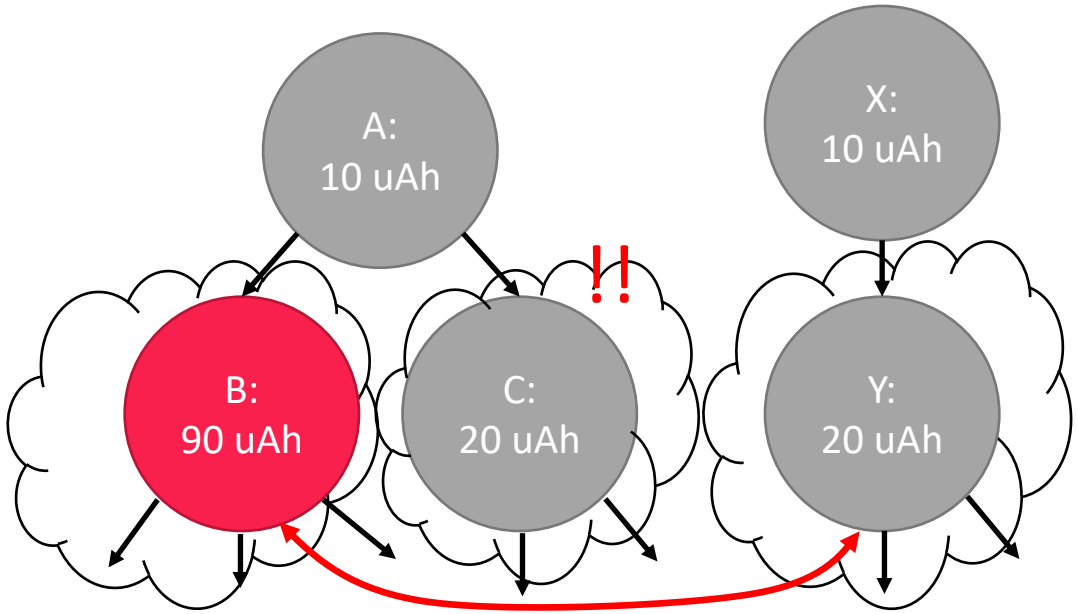
Part II: How to diff?

What should be the diffing granularity?

How to perform diffing?

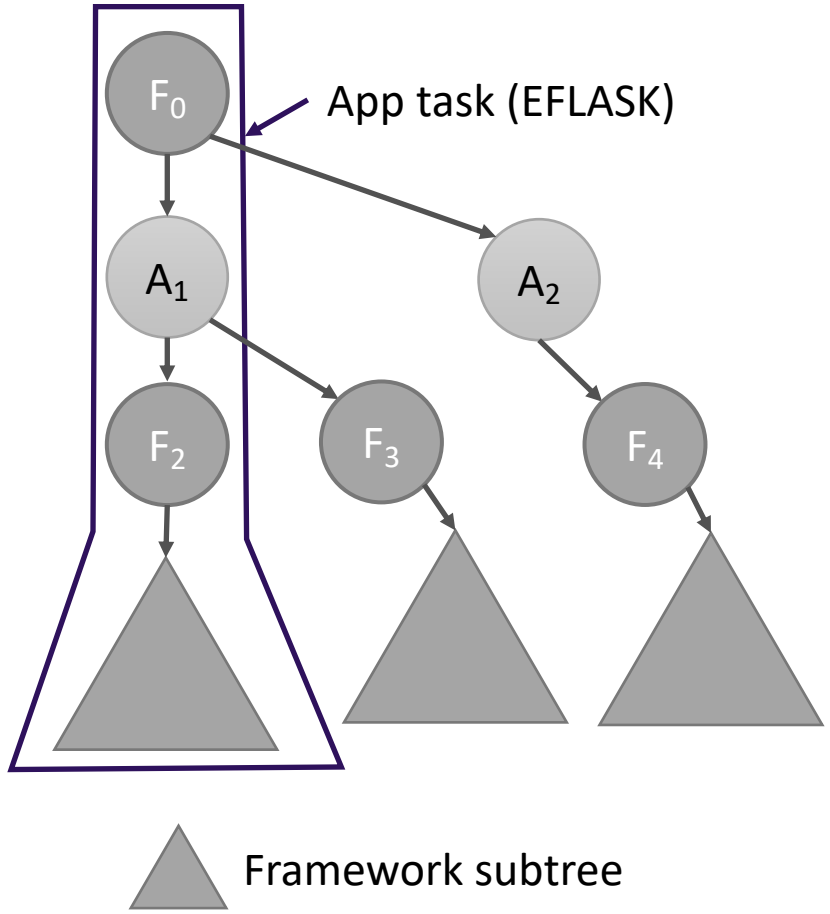
What should be the diffing granularity?

- Similar apps perform similar core *tasks*
  - Music app performs music playback, UI updates such as progress bar, text boxes
- Diffing should be performed on app tasks



How to identify app tasks from energy profile?

- App tasks manifest as EFLASK (Erlenmeyer flask shaped)
  - Call path
  - Neck F-method
  - Subtree
- Identifying matching tasks boils down to matching EFLASK



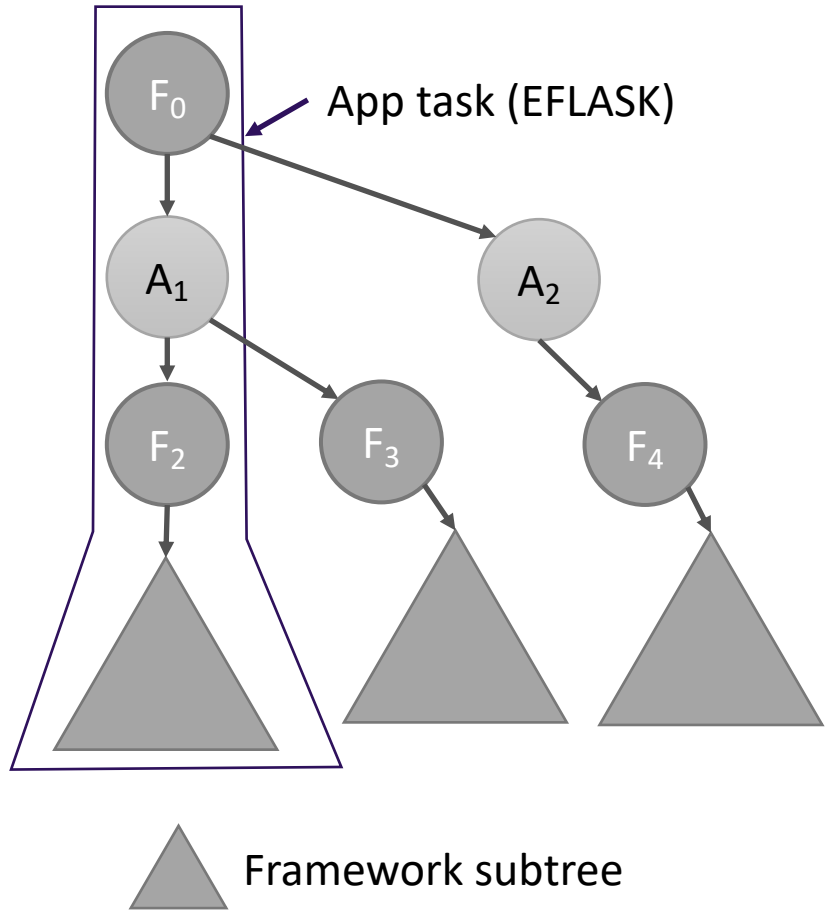
What tree structures to diff?

Tree type	Size	Path preserving
Call tree	O(millions)	Yes
Dynamic call graph	O(thousands)	No
Calling context tree	O(ten thousands)	Yes

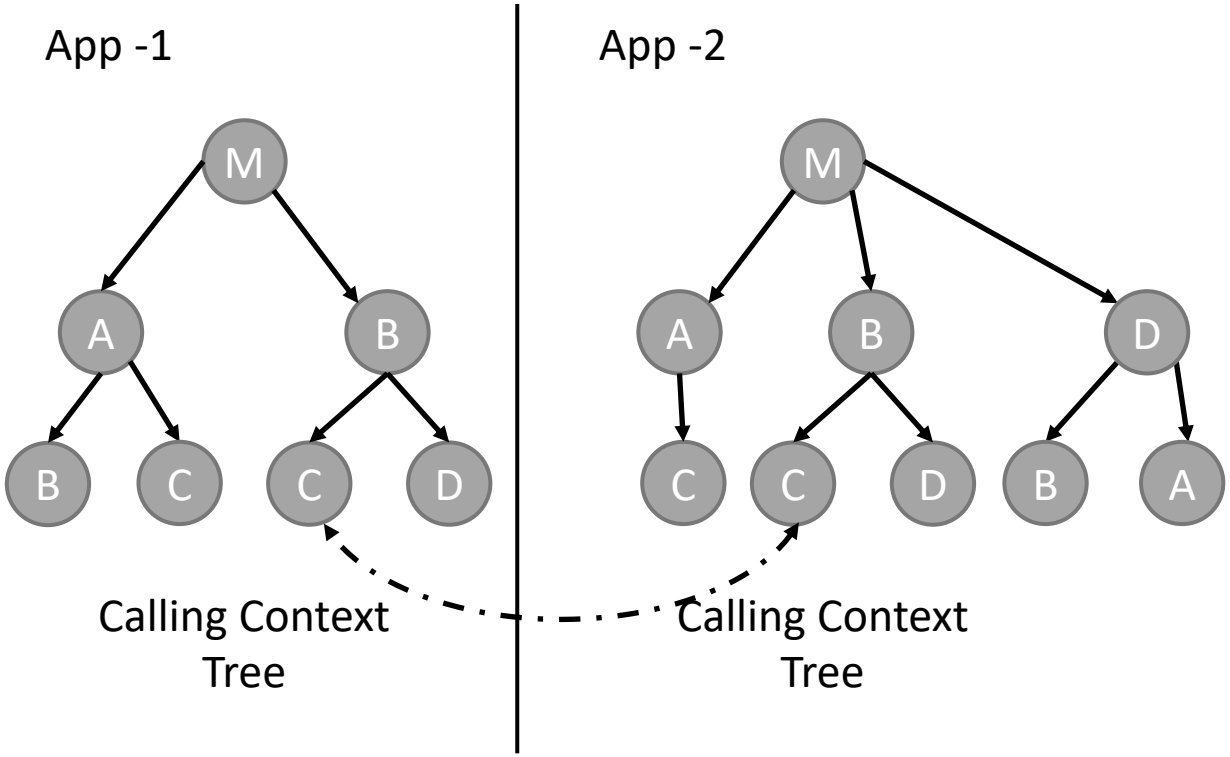
How to perform diffing?

- Previous tree matching algorithms are not applicable

- EFLASK matching algorithm
  - Simultaneously identifies EFLASKs and finds matching EFLASKs between two similar apps



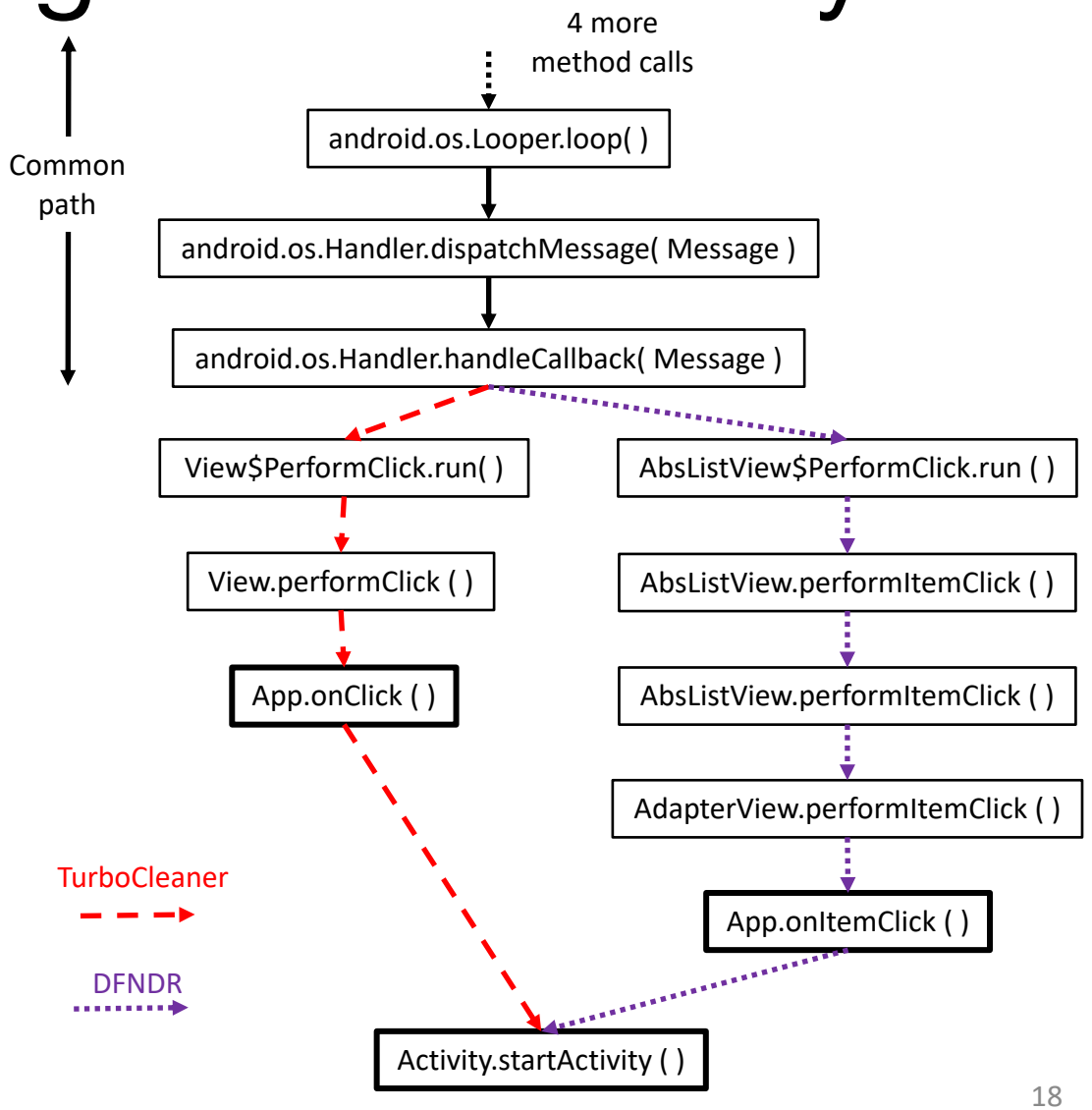
Exact path matching



- Unique node with same path from root

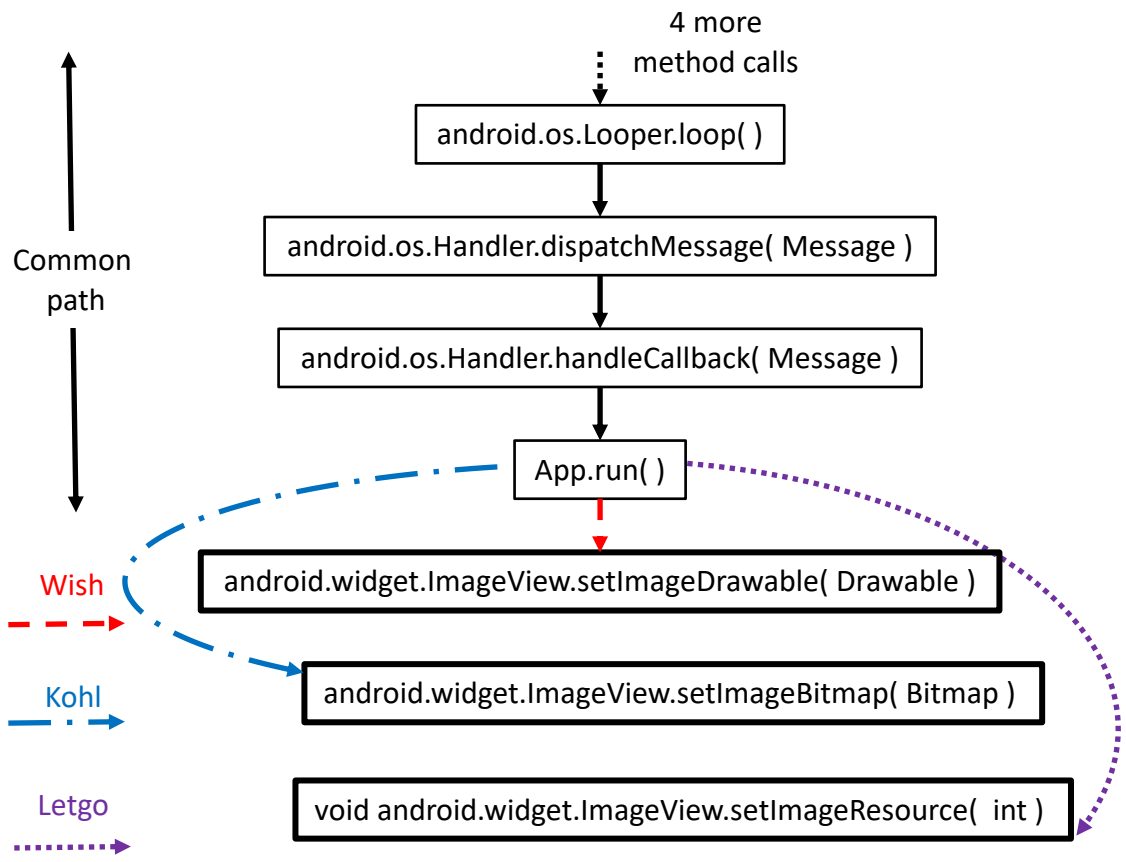
EFLASKs for matching tasks can vary

- **Call paths can vary slightly**
  - Use different mechanism to get same callback
  - Use different callbacks to receive similar events



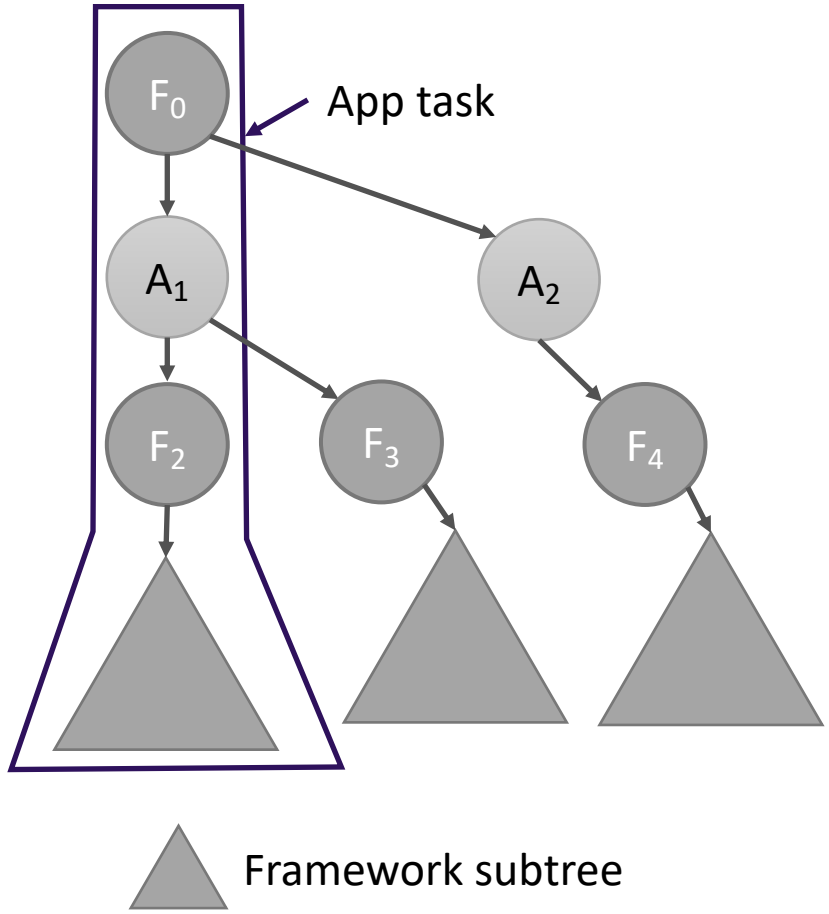
EFLASKs for matching tasks can vary (2)

- Call paths can differ slightly
- **Neck F-methods may vary**
  - Use different classes that implement same APIs
    - HttpConnectionURLImpl, HttpsConnectionURLImpl
  - Use alternate APIs to perform same task



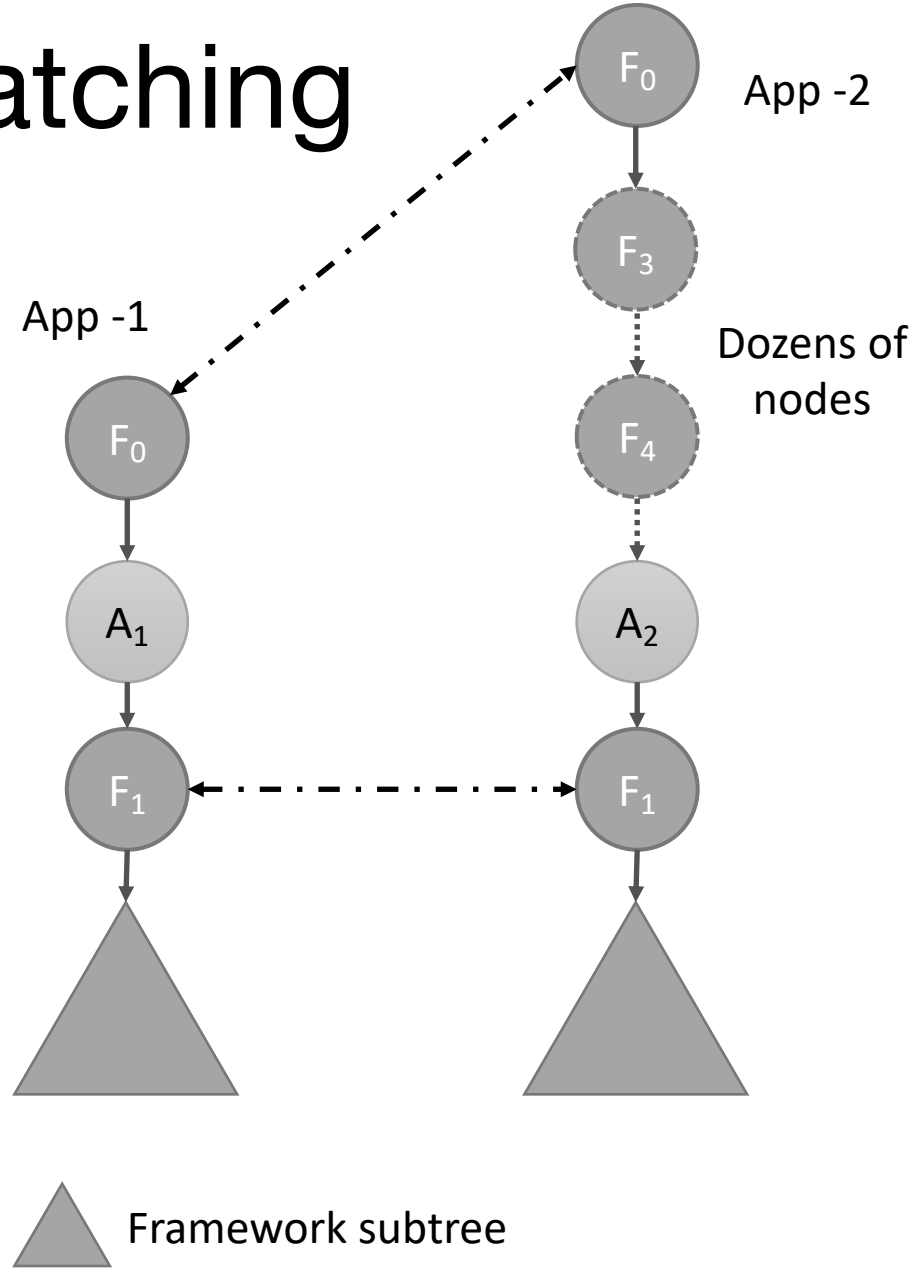
EFLASKs for matching tasks can vary (3)

- Call paths can differ slightly
- Neck F-methods may differ
- **F-method subtrees may vary**
  - Program state, call parameters determine F-method subtree



Prior approximate tree matching algorithms

- [Zhang *et.al.* Algorithmica 1995] produces maximal matching
- Drawback: matches EFLASKS with arbitrarily different paths
  - Maximize subtree overlap, disregard paths



EFLASK matching algorithm

Algorithm	Passes	Approach	Drawback
Exact Path Matching	Top-down	Matches paths, disregards subtree	Can't handle path variations
Approximate Tree Matching (Zhang et.al)	Bottom-up	Maximizes subtree overlap, disregards path	Matches nodes with arbitrarily different paths
EFLASK matching algorithm	1 top-down, 1 bottom-up pass	Maximizes subtree overlap while respecting path similarity	

EFLASK Matching Algorithm

$$\forall\,(v,w) \in M : w \in C_\alpha(v)$$

1. Top down pass: Calculate $C_\alpha(v)$

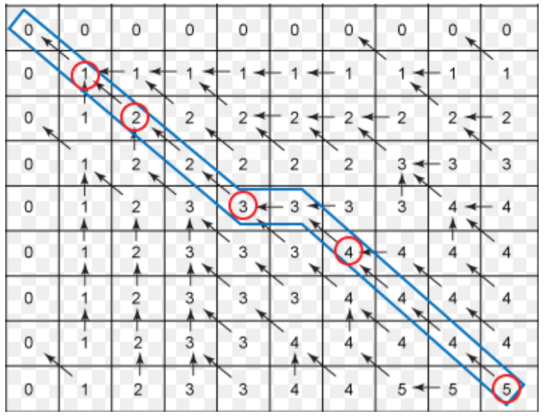
$$C_\alpha(v) = \{w \in V(T_2) \mid \rho(s(v),s(w)) \leq \alpha\}$$

2. Bottom up pass: Calculate $\mu_\alpha(v)$

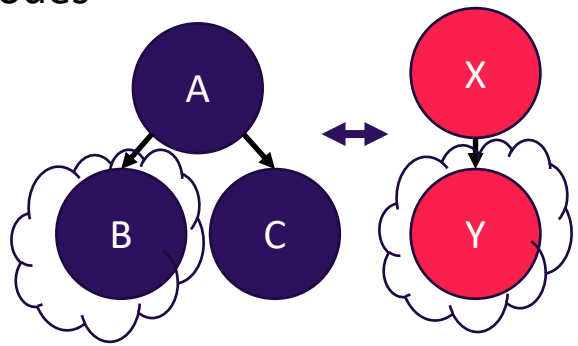
$$\mu_\alpha(T_1(v),\theta) = 0$$
$$\mu_\alpha(\theta,T_2(w)) = 0$$
$$\mu_\alpha(T_1(v),T_2(w)) = 0 \qquad if\ w \notin C_\alpha(v)$$

$$\mu_\alpha(T_1(v),T_2(w)) = max\left(\begin{array}{l} \max\limits_{w_j \in child(w)} \mu_\alpha(T_1(v),T_2(w_j)) \\ \max\limits_{v_i \in child(v)} \mu_\alpha(T_1(v_i),T_2(w)) \\ \max\limits_{RM(v,w)} \mu_\alpha(RM(v,w)) \\ \qquad +(1-\gamma(v,w)) \\ otherwise \end{array}\right);$$

3. Use backtracking to find matched nodes

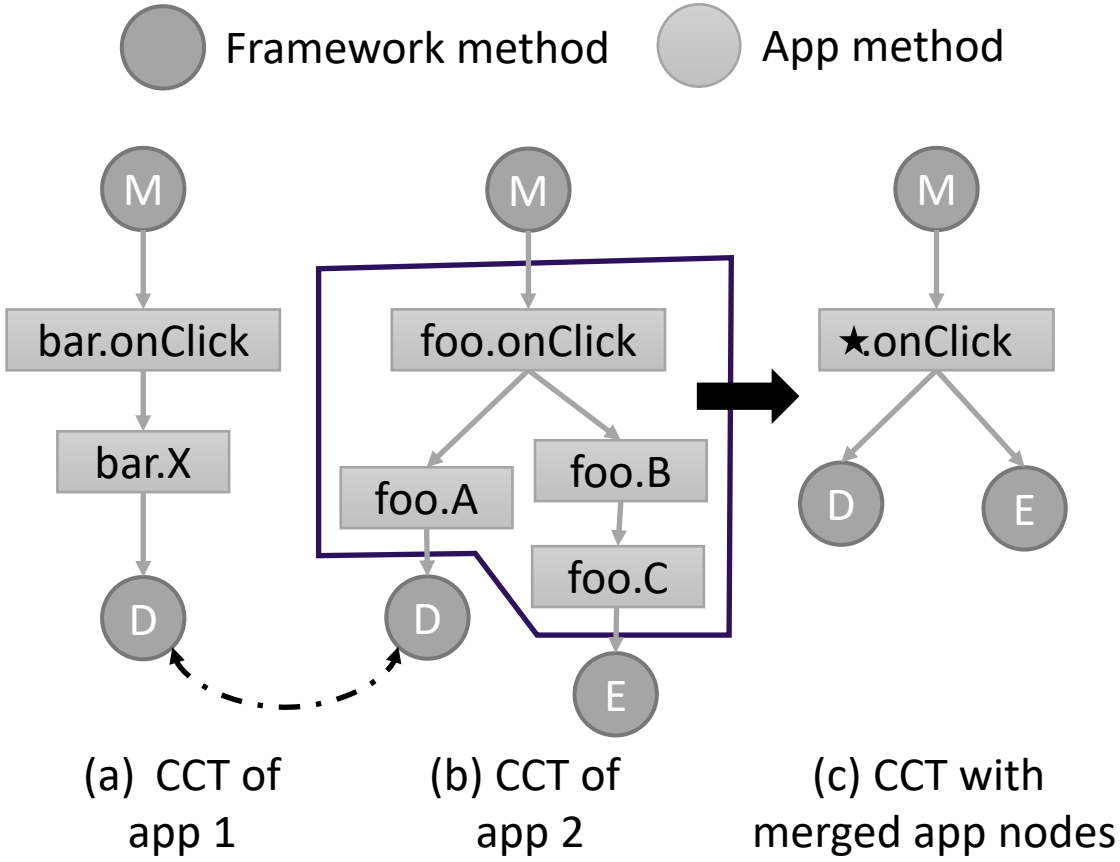


4. Finds matching EFLASKS based on maximally matched nodes



Reducing unimportant call path variations – Collapsing app methods

- Internal app method names are arbitrary and often obfuscated
- App callback method names are well-defined by framework
  - foo.onClick overrides onClickListener.onClick



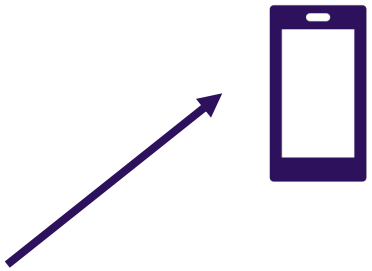
Part III: It works!

DIFFPROF implementation

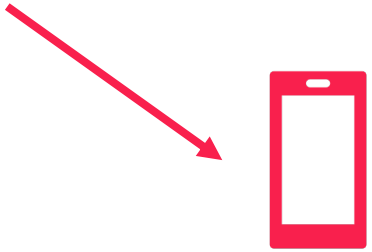
- Built on top of Eprof [Pathak *et. al.* EuroSys' 12]
- Diffing and GUI front-end
  - 5.7k lines Java code

Developer workflow with DIFFPROF

Write automated tests for two similar apps

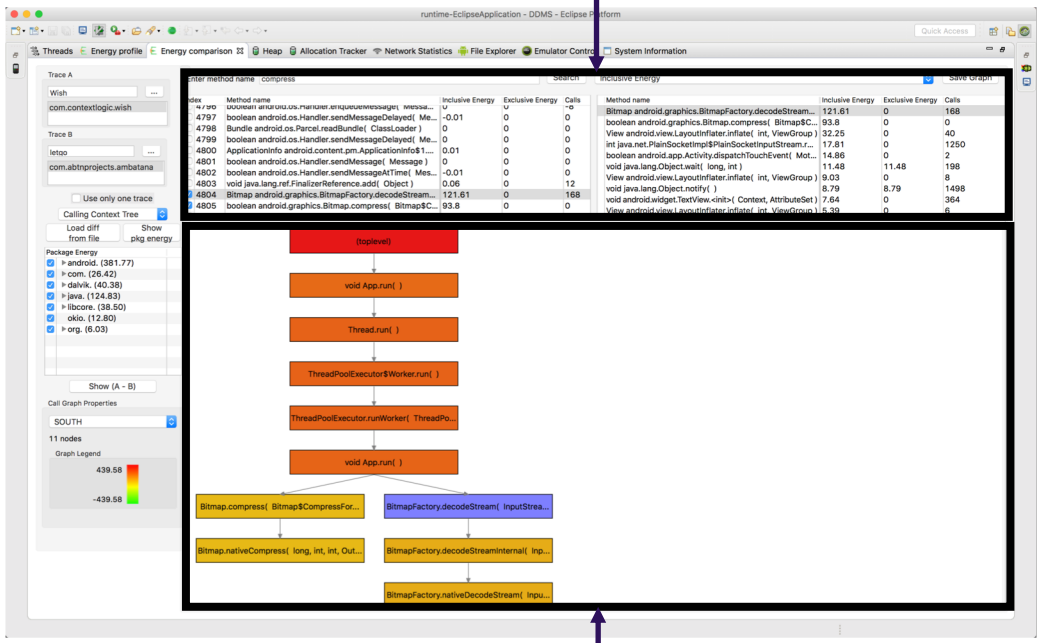


Upload energy profiles to DIFFPROF



Run the tests with Eprof and collect energy profiles

Tabular output



Graphical output

Evaluation

- Android's UI Automator tests
- 8 app groups- 5 popular apps, 5 versions of one app, majority with 50M+ installs

App Category	App group	Similar/Competing Apps
Communication	Instant Messaging	Whatsapp, Google Hangouts, Facebook Messenger, Line, TextNow
	Email	Android mail, Aqua Mail, Email For Any, MailRU, myMail
Music & Audio	Music streaming	Spotify, Pandora, Soundcloud, iHeartRadio, Free music
Personalization	Launcher	GO, CM Launcher 3D, APUS, Solo, Hola
Productivity	File explorer	ES, FX, Solid, File explorer, File manager
Shopping	Shopping	Wish, eBay, Amazon, Kohl, letgo
Tools	Antivirus	CM Security, AVG, DU, Mobile Security & Antivirus, Kaspersky
	Cleaning	Clean Master, DFNDR, Fast Cleaner, Turbo cleaner, DU, Ccleaner

Evaluation – Matching task statistics

App	Matched tasks' energy
Antivirus	
AVG	92.73%
CMSecurity	79.64%
DU	85.90%
Kaspersky	73.99%
MobileSec	69.98%
Cleaner	
CCleaner	76.57%
Clean Master	70.82%
DFNDR	73.52%
Fast Cleaner	94.89%
Turbo Cleaner	88.46%
…	
Average	78.86%

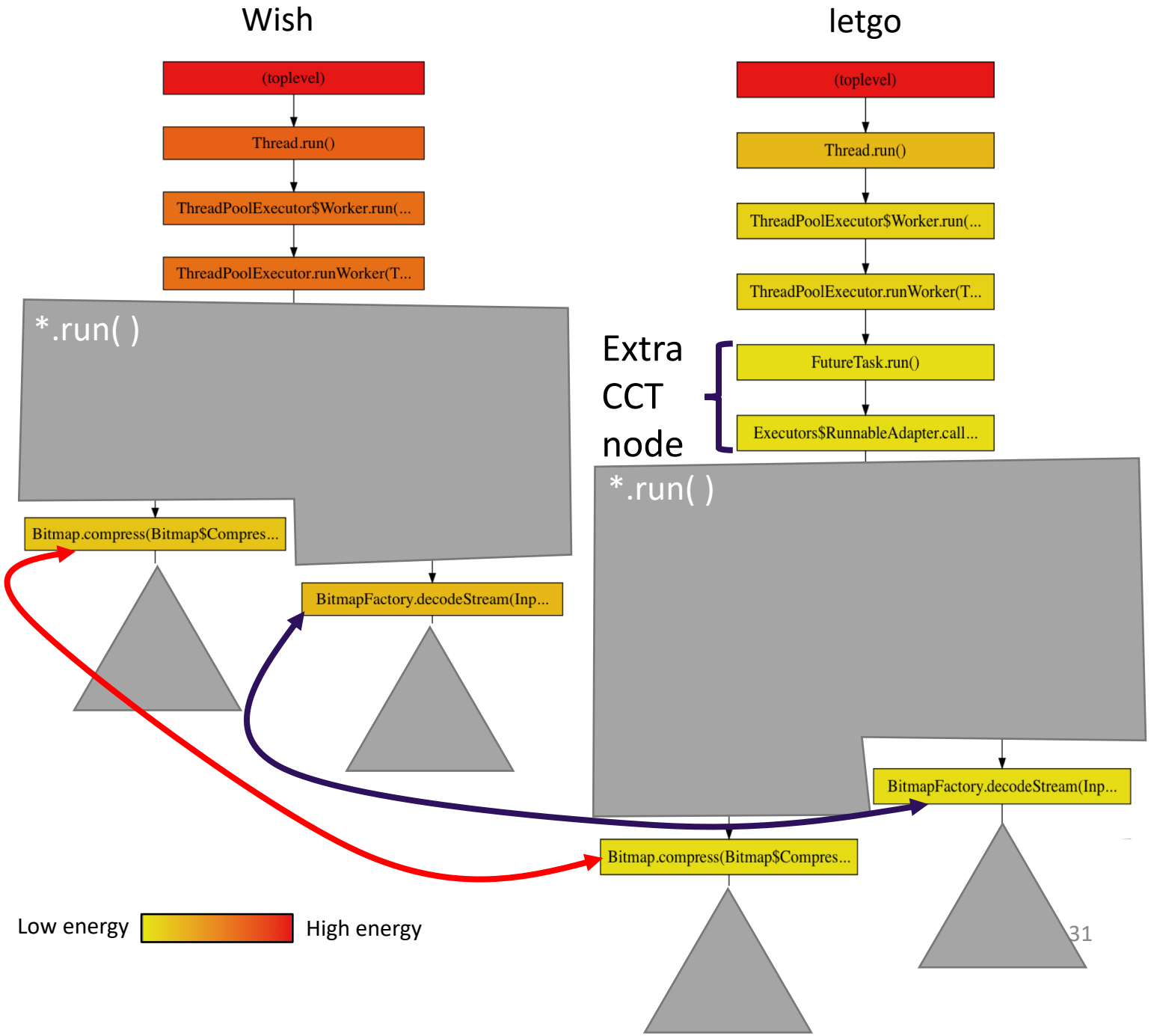
Evaluation – Case studies

- Found 12 energy optimizations in 9 popular apps
  - 3 of them already confirmed by developers
  - Saves 5.2% - 27.4%

App	Task	Percentage of total energy drain
Wish letgo	Bitmap.compress	15.9% 3.6%
Wish letgo	BitmapFactory.decodeStream	19.9% 2.5%
Pandora5.7 Pandora8.3	TextView.setText	28.1% 0.7%
Spotify Pandora	ProgressBar.setProgress	20.2% 1.6%

EFLASK matching algorithm's effectiveness:

Wish vs letgo



DIFFPROF vs Eprof: Wish vs letgo

DIFFPROF

Task energy drain difference rank	Task Name	Wish energy drain (µAh)	Letgo energy drain (µAh)
1	BitmapFactory .decodeStream	126.3	5.01
2	Bitmap.compress	100.9	7.14
3	LayoutInflater .inflate	62.46	17.03

Eprof

Method energy drain rank	Method name	Wish Energy drain (µAh)
1	Thread.run	395.0
2	ThreadPoolExecutor .runWorker	353.9
	…	
18	BitmapFactory.decodeStream	126.3
28	Bitmap.compress	100.9

Wish vs letgo : Energy optimization

- Setting breakpoint at Bitmap.compress method reveals
  - Wish uses *png* images with quality set to *100*
  - letgo uses *jpg* images with quality set to *90*

Conclusions

Poster# 46

Why diffing?

- Dozens of similar apps
- Similar apps differ significantly in battery drain
- Framework services dominate battery drain

How to diff?

- App tasks manifest as EFLASK (call path, neck F-method, subtree)
- EFLASK matching algorithm

Diffing works!

- DIFFPROF matches tasks consuming 80% of total energy
- Found 12 energy optimization opportunities in 9 popular apps