Summary:

Objective:

1. Analysis of the activities of various apps running in the background on thousands of phones in the wild.
2. Quantify the amount of battery drain by all such background activities and possible energy saving.
3. Develop a metric, to measure the usefulness of background activities, that is personalized for each user.
4. Devise HUSH (screen-off optimizer) that monitors the above-mentioned metrics online and automatically identifies and suppresses background activities during screen-off periods that are not useful to the user.

Hypothesis:

1. background activities of apps are meant to improve user app experience but they are only useful if the user interacts with those apps in the foreground sometime during the next screen-on interval;
2. the usefulness of background activities of an app is likely to be user-dependent and thus their occurrences should be personalized.

Existing :

1. iOS: "Disable Background Refresh" which basically prevents the apps from running in the background altogether.
2. Android: "Restrict Background Data" which prevents the user from running up charges on their mobile data usage due to background data activities by switching background data refresh to using WiFi only.

Limitation:

disable background activities of all apps and hence do not distinguish background activities of apps that are potentially more useful to the user from the rest, which can adversely affect user experience.

Detailed Summary:

Methodology:

* Break down CPU time of a device, into the following components:

CPU idle time during screen-off;

CPU busy time (background apps and services) during screen-off;

CPU idle time during screen-on;

CPU busy time (for all apps and services) during screen-on.

* Total energy per day per device among different activities as follows:

Energy by WiFi beacon, WiFi scanning, cellular paging, and SOC

base power during screen-off;

Energy by CPU idle during screen-off;

Energy by background services and apps during screen-off;

Energy by background services and apps during screen-on;

Screen energy during screen-on.

Energy by CPU idle during screen-on;

Energy by foreground apps excluding screen.

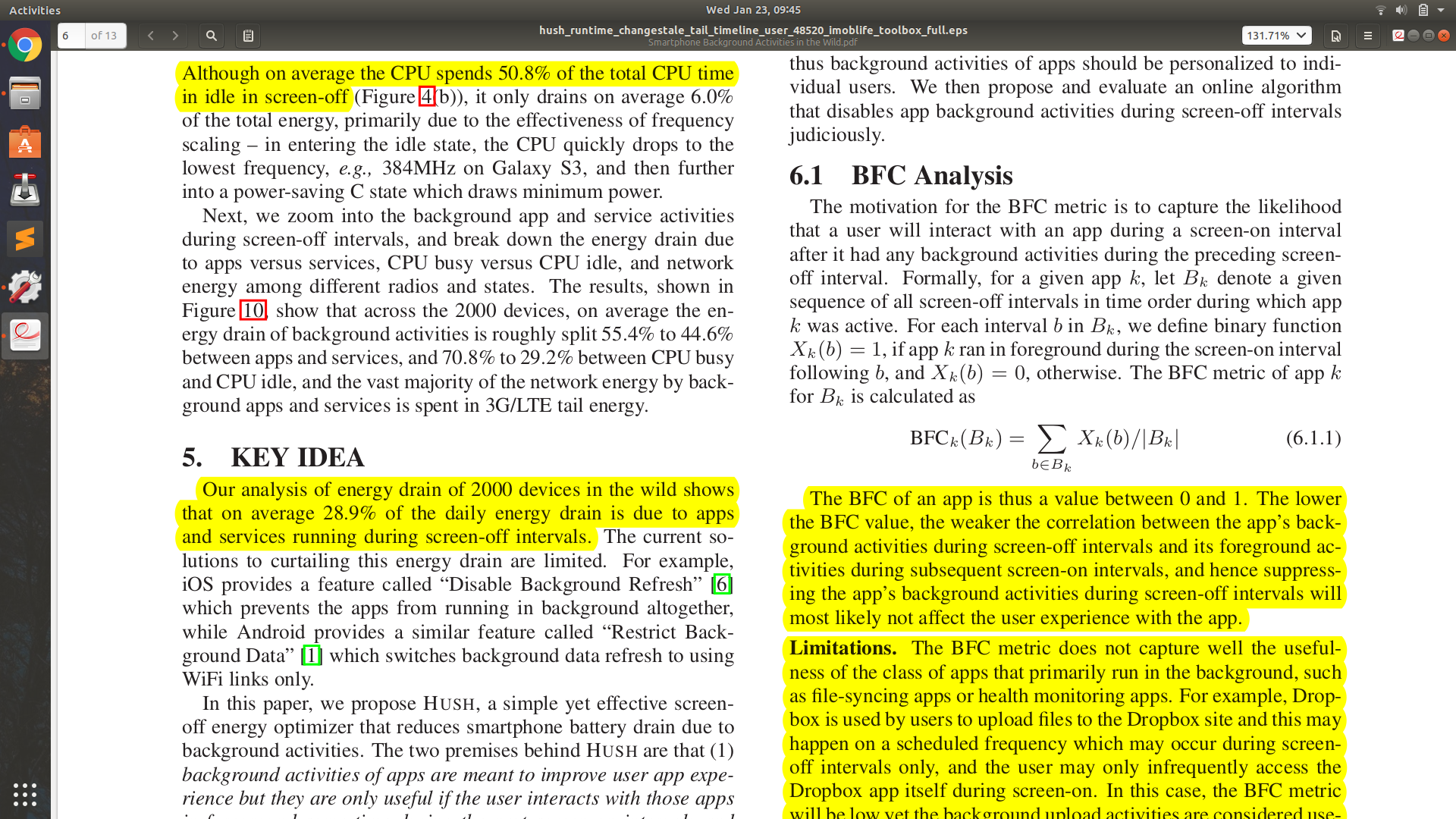
* HUSH learns the usefulness of apps’ background activities during screen-off intervals, it instructs the Android scheduler to selectively avoid scheduling the execution of certain apps during screen-off intervals to save the screen-off energy.
* BFC Analysis: Captures the likelihood of interaction with an app during a screen-on interval after it had any background activities during the preceding screen-off interval.
* k: given app

: the sequence of all screen-off timely in which was active

: binary function if was

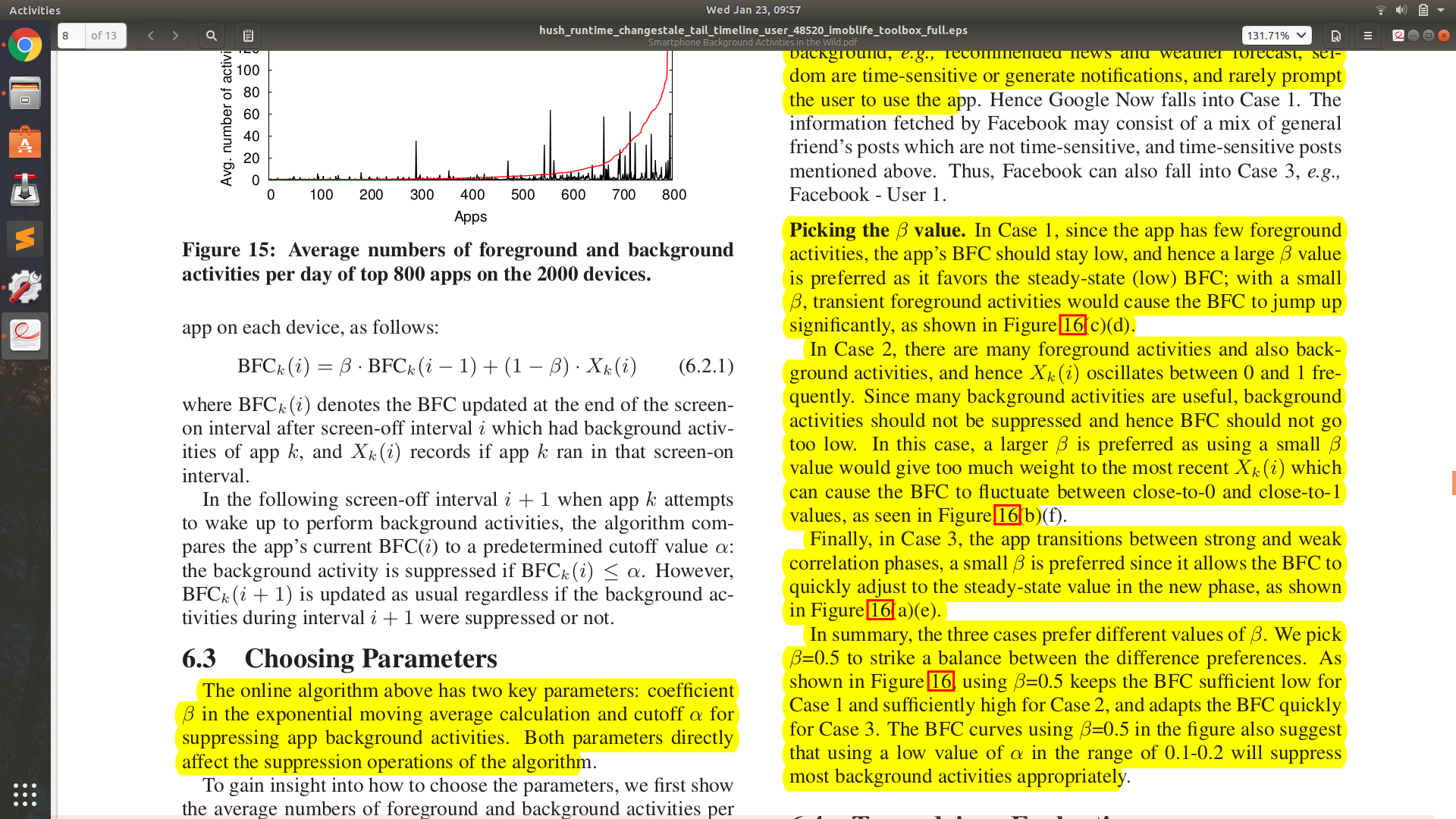
1: active in the consequent screen on

0: else



* The BFC lies between 0 and 1. The lower the BFC value, the weaker the correlation between the app’s background activities during screen-off intervals and its foreground activities during subsequent screen-on intervals, and hence more likely to be suppressing the app’s background activities during screen-off intervals.

Limitations. The BFC metric does not capture well the usefulness of the class of apps that primarily run in the background, such as file-syncing apps or health monitoring apps.

* Prediction based online algorithm: 

α: threshold param

β: in the exponential moving average calculation

* β approximation:

Case 1: Higher since switching would cause BFC to jump otherwise.

Case 2: Higher since otherwise, the value would give too much weight to the most recent which can cause the BFC to fluctuate.

Case 3: a small β is preferred since it allows the BFC to quickly adjust to the steady-state value.

* α approximation: Energy saving. Evaluate the algorithm with different threshold α values, which control the trade-off between screen-off energy reduction with app staleness.
* Trace-driven Evaluation: The energy saving is defined as and are the total energy on each device before and after our suppression algorithm is turned on.
* When an app background activity is suppressed, the associated service, networking, and CPU idle energy drain are also avoided.
* Staleness: To quantify mispredictions( Case 1 especially) due to past BFC we use elapsed time since the last background activity, or the last foreground activity of the app, whichever is closer in time. The staleness of an app on a device is then defined as the average staleness of all its foreground activities on that device.
* HUSH does not maintain the BFC calculation instead, it simply maintains and tunes a single threshold time parameter (τ ) for each app; an app’s next screen-off activity is allowed only if its previous background activity happened τ earlier, and is otherwise suppressed. Whenever HUSH allows a screen-off activity, τ is multiplied by a scaling factor σ (>1), making screen-off activities less and less frequent. However, when an app comes to the foreground, its τ is reset back to τinit.

Experiment:

* 2000 Galaxy S3 and S4 devices.
* Designed a free Android app called eStar installed on a user’s phone, performs periodic logging of the usage of various phone components by apps and system services.
* Verified using their own S3 and S4 that CPU was always fully awake in screen-off intervals when phones were being charged via USB.
* Removed all the USB charging intervals and the 6-second CPU linger time from all the rest screen-off intervals.
* Flag the apps as suspicious apps per device to later classify apps as “no-sleep” or not per app.
* Measuring energy drain is performed by developing a power model using triggers deploying an app that collects all these triggers for the power model.
* Stability of BFC: For each user trace, divide the trace duration into non-overlapping windows of 24 hours each, and calculate the BFC for each window for each app that has at least 24 daily background activities.

Results:

* Screen-off intervals tend to last much longer than screen-on intervals.
* No significant correlation between the total app busy time and the total CPU idle time
* Primary suspect for causing the large fractions of CPU idle time during screen-off intervals is an inefficient or incorrect use of wakelocks in apps.
* On average 45.9% of the total energy drain in a day occurs during screen-off periods.
* The background apps and services during screen-off including the induced CPU idle together contribute to 28.9% of the total energy drain.
* BFC of the apps of the same user varies: 60% of the apps have a BFC value of 0, suggesting that many apps are not accessed at all by users after they performed some background activities.
* BFC of an app varies with users.
* BFC for the same app on the same device is fairly steady.
* The average numbers of foreground and background activities per

day of the top 800 apps on the 2000 devices :

22% of apps have more than 10 background activities per day

10% of apps have more than 36 background activities per day,

77% of apps have more daily background activities than daily foreground activities.

* Apps can be grouped based on background activity patterns:

Case 1: Few foreground activities and hence many background activities per foreground activity. Rarely prompt the users to the app (e.g., Google Now, and Gmail);

Case 2: Many foreground activities and hence few background activities per foreground activity. Notifies the user and is most likely to switch (e.g., Facebook-User 2, Whatsapp-User 2);

Case 3: Alternating phases with few and many foreground activities (e.g., Facebook-User 1, Whatsapp-User 1).

* Most of the energy saving comes from Case 1 apps, whose BFC will almost always stay below 0.1
* The BFC-based suppression algorithm strictly follows an all-or-none policy; once the BFC decays below α, it suppresses all background activities in the subsequent screen-off intervals leaving the staleness to grow boundlessly until the next foreground activity.
* HUSH works well not only for Case 1 app but also for Case 2.

HUSH Design

* The main algorithm of HUSH is implemented inside BatteryStatsImpl, where Android maintains runtime statistics of the whole system. Other HUSH components directly interact with the theBatteryStatsImpl module.
* HUSH tracks app foreground and background activities and intercepts app invocations by framework services during screen-off intervals by either allowing or rejecting them.
* allowHush allows all requests when the screen is on and when an app is perceptible to the user in screen-off, such as music streaming apps and navigation apps. Users can also whitelist apps and vary the aggressiveness of HUSH by changing the decay parameter σ in Android settings.
* On average, HUSH suppressed 4400 and 5543 app background activities on the two phones daily, which reduced the daily CPU busy time by 1.69x and 2.23x.
* HUSH significantly reduced the average screen-off power, i.e., screen-off energy normalized by screen-off duration, by 2.95x and 1.46x, while the average screen-on power is slightly increased, leading to an overall average power reduction of 1.25x and 1.44x for the two users, respectively.

Positive Points:

1. Released Source Code.

The HUSH source code was released in March 2015 and can be downloaded at github.com/hushnymous.

1. Handle corner cases like Staleness (Though examples not provided).
2. A number of Experiment conducted on daily usage, energy usage, and drain, background/foreground activities, CPU idle/busy time etc before taking decisions like approximation values.

Negative Points:

1. Explained in detail about BFC but quickly discard it as a flaw.
2. No detailed explanation on the actually used algorithm behind HUSH.
3. Vague explanation on how HUSH implementation and procedure to actually suppress the background app.
4. Lack of diversity only tested on one kind of phone(Samsung S3/S4) the whole time.
5. Did not explain about the overhead due to HUSH itself, only mentioned how much it affected the result.