Pre-fetch Procrastinator for Android Apps

The following are the team members for the project:

1. Akshay Kamath
2. Amish Shah

What is the problem you are trying to solve?

Prefetching content is a methodology that developers use to download the content even before a user needs it. The primary benefit of prefetching content is that the app becomes more responsive. For instance, if there is a weather app and along with the homepage at the launch of the app, all the articles and images in other Views are also prefetched. So, if the user wants to visit different Views inside the app, the content is rendered quickly. However, if the user usually works with the homepage and does not visit the other Activities/Views than the data consumed to prefetch content is wasted. To infuriate things further, if a user has enrolled for a pay-per-byte plan, or he/she is near the end of billing cycle or currently under a roaming plan, the user may incur monetary loss due to pre-fetch.

*What are the key challenges?*

*Deployment, such a work has never been done on Android platform*

* *How to identify network calls for images in particular and store them*
* *Callback methods*
* *Group applications in terms if patterns*
* *We don’t have data sense, think of an algorithm or leverage something that functions and get those 3 cases or create a new algorithm*
* *Checking the visibility of UI elements – activity by activity bases* *once the UI element is found retrieve them*
* *Compare change in n/w and user feedback*

What is the high-level approach you plan to take?

We first plan to create an app and inject some prevalent prefetching techniques. Alongside, we plan to modify the algorithm mentioned in [2] that finds the correlation between off-screen background activities of an app and the user interacting with that app later during screen-on – BFC (Background to Foreground correlation) metric. For our work, in a similar fashion, we would like to capture the likelihood that the user will interact with a particular Activity in an app depending on the times the user interacted with that activity during previous launches.

As per the algorithm, we would track the user movements in the activity until we calculate a confident correlation (prefetch score) among activities. We will also maintain a UI-Call map for a View that will contain all the UI elements with their respective pre-fetch network calls.

Once we calculate a prefetch score, when an app or a new view is launched, we traverse the respective UI-Call map and delay the calls for those activities that have prefetch scores lower than the threshold. When the delayed UI elements/Views are visited, we execute the network calls for those elements.

*Expected results:*

1. *We definitely changes relevant to network usage statistics*
2. *Users will evaluate by playing with the application*

RoadMap

1. *Week 1 and 2: Create/Search an app and improvise algorithm*
2. *Week 3: Research patterns and maintain the global elements for restoration*
3. *Week 4: Design an algorithm*
4. *Week 5: Based upon the prefetch policy retrieve if UI elements appear.*
5. *Week 6: Improve the scalability to include more elements for prefetch tasks*
6. *Week 7: Inject in real world applications*
7. *Week 8:*
8. *Week 9: Evaluation threshold various threshold values*
9. *Week 10: user feedback, report and analysis*