TextNow Android Memory Analysis with ADM and Eclipse Memory Analyzer

Procedure

LINK : https://www.linkedin.com/pulse/fixing-memory-leaks-android-studio-albert-lai

Developers

Xyan Bhatnagar

Explanation

OutOfMemoryExceptions arise from when your app has been leaking memory gradually until the OS decided enough was enough and killed your app at some pseudo-arbitrary moment.

The Java Language is Garbage Collected, so memory management isn't a cause of concern most of the time on Android.

But, the Garbage Collector can only remove objects that are no longer "reachable" via something called the GC root.

Leaks result from static references to objects we no longer need, e.g. Activities we thought were destroyed.

Analysis

Analysis #1

Date : June 30th, 2016

Branch : develop

Version : 4.15.0

Testing Conditions

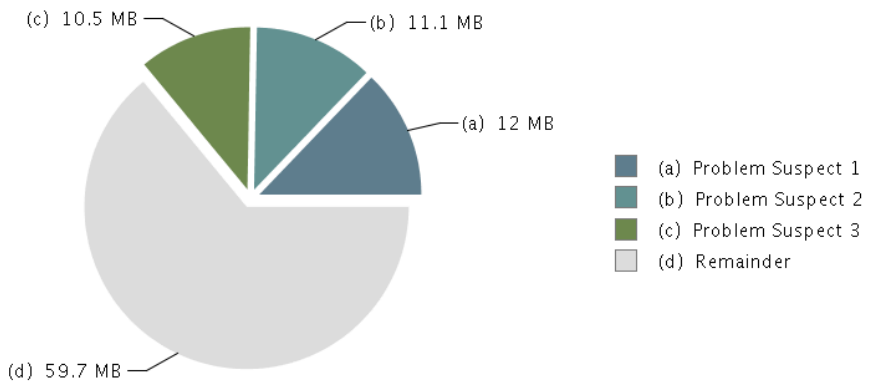
1. Nexus 5 running Android 6.0 Marshmallow
2. Fully Charged Device
3. WiFi enabled
4. No Ads (Premium Account)
5. Continuously moving between Conversation List and Message List (roughly 100 times)

Results

Heap Allocation spiked with each successive move between activities. Suspected Memory Leaks.

Leaks

Overview



Total: 93.3 MB

Was able to expand the heap to roughly 5x simply by moving between Conversation List and Message List.

INITIAL (After Application reached Conversation List)

Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	45.442 MB	41.793 MB	3.649 MB	91.97%	214,356

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	40,338	7.040 MB	16 B	508.797 KB	48 B	183 B
data object	162,525	8.919 MB	16 B	37.219 KB	32 B	57 B
class object	3,446	1.910 MB	144 B	8.000 KB	464 B	581 B
1-byte array (byte[], boolean[])	990	26.382 MB	16 B	7.119 MB	64 B	27.287 KB
2-byte array (short[], char[])	217	131.305 KB	16 B	51.156 KB	48 B	619 B
4-byte array (object[], int[], float[])	23,784	3.538 MB	16 B	300.000 KB	64 B	155 B
8-byte array (long[], double[])	474	21.750 KB	16 B	2.000 KB	32 B	46 B
non-Java object	2	504 B	24 B	480 B	480 B	252 B

FINAL (After moving between Conversation List and Individual Conversations)

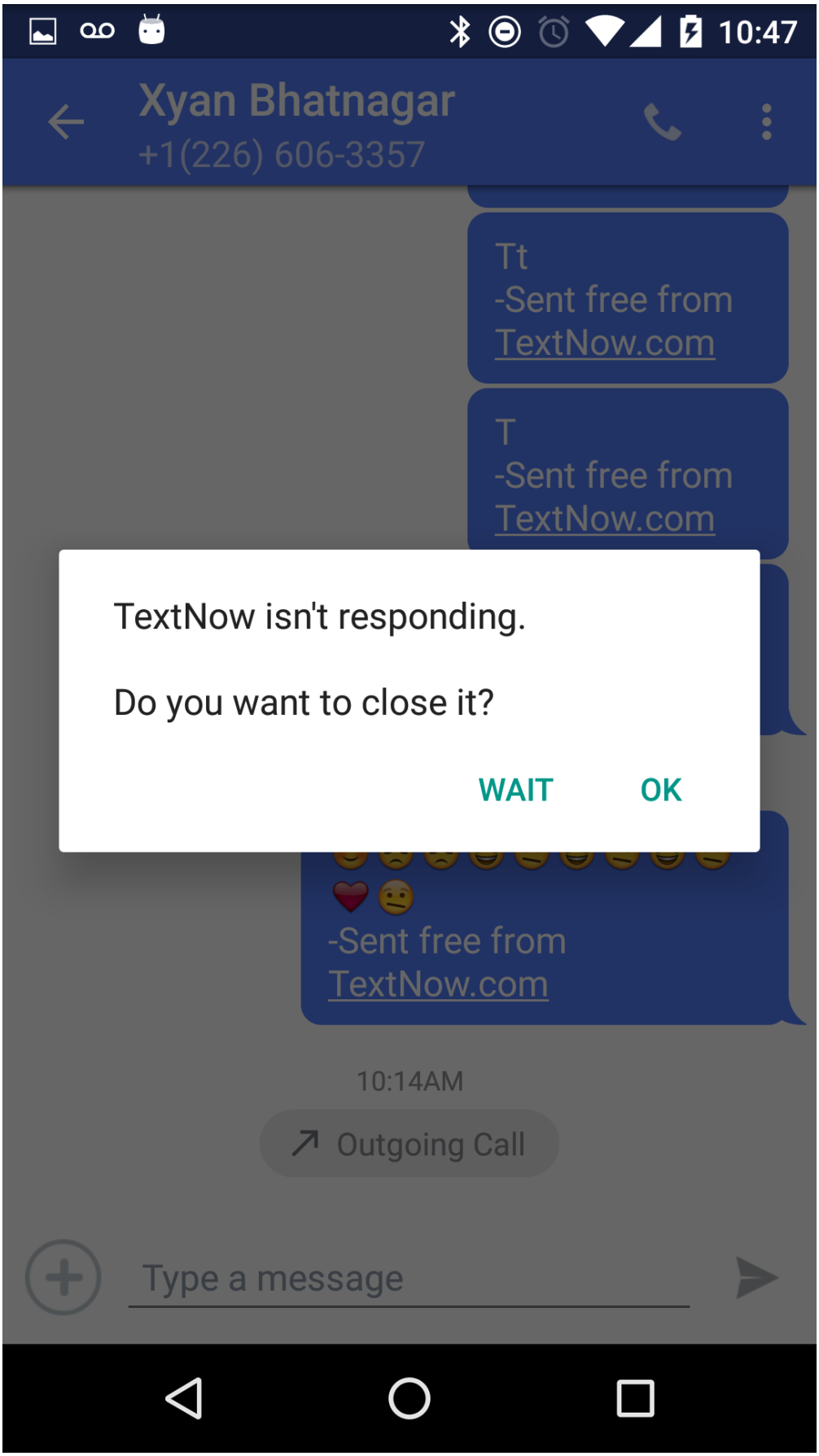
Heap updates will happen after every GC for this client

ID	Heap Size	Allocated	Free	% Used	# Objects
1	194.943 MB	178.943 MB	16.000 MB	91.79%	1,214,385

Cause GC

Display: Stats

Type	Count	Total Size	Smallest	Largest	Median	Average
free	57,335	11.562 MB	16 B	52.000 KB	64 B	211 B
data object	975,296	64.386 MB	16 B	37.219 KB	32 B	69 B
class object	3,946	2.237 MB	144 B	8.000 KB	464 B	594 B
1-byte array (byte[], boolean[])	5,906	93.490 MB	16 B	7.119 MB	48 B	16.209 KB
2-byte array (short[], char[])	1,770	301.961 KB	16 B	51.156 KB	32 B	174 B
4-byte array (object[], int[], float[])	185,012	14.041 MB	16 B	300.000 KB	48 B	79 B
8-byte array (long[], double[])	19,539	645.453 KB	16 B	2.000 KB	32 B	33 B
non-Java object	2	504 B	24 B	480 B	480 B	252 B


10:47
Xyan Bhatnagar
+1(226) 606-3357
Tt
-Sent free from
TextNow.com
T
-Sent free from
TextNow.com
TextNow isn't responding.
Do you want to close it?
WAIT
OK
-Sent free from
TextNow.com
10:14AM
Outgoing Call
Type a message

Possible Suspects

1. **Leanplum**
    a. Suspected due to Android Resources held by Class
    b. Occupies roughly 12% of overall heap allocation

> 7,709 instances of **"java.lang.Class"**, loaded by **"<system class loader>"** occupy **12,547,488 (12.83%)** bytes.
>
> Biggest instances:
>
> - class android.content.res.Resources @ 0x70063d28 - 9,225,816 (9.43%) bytes.
> - class com.leanplum.aZ @ 0x32d70c00 - 1,486,672 (1.52%) bytes.
>
> **Keywords**
> java.lang.Class
> Details »

2. **TintedImageView**
    a. Suspected due to Drawer. Found references to mHome, mWirelessStore, etc.
    b. Occupies roughly 12% of overall heap allocation

> 223 instances of **"com.enflick.android.TextNow.views.TintedImageView"**, loaded by **"dalvik.system.PathClassLoader @ 0x32c02ce0"** occupy **11,636,512 (11.90%)** bytes.
>
> Biggest instances:
>
> - com.enflick.android.TextNow.views.TintedImageView @ 0x32ec4200 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x3360d800 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x33d8be00 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x342fd400 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x34513800 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x34a33a00 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x34e15c00 - 1,153,256 (1.18%) bytes.
> - com.enflick.android.TextNow.views.TintedImageView @ 0x34f17000 - 1,153,256 (1.18%) bytes.
>
> These instances are referenced from one instance of **"java.lang.Object[]"**, loaded by **"<system class loader>"**
>
> **Keywords**
> java.lang.Object[]
> com.enflick.android.TextNow.views.TintedImageView
> dalvik.system.PathClassLoader @ 0x32c02ce0
> Details »

3. **AvatarView**
    a. Suspected due to Contact Avatar in ConversationAdapter.
    b. Occupies roughly 12% of overall heap allocation

123 instances of **"com.enflick.android.TextNow.views.AvatarView"**, loaded by **"dalvik.system.PathClassLoader @ 0x32c02ce0"** occupy **11,023,768 (11.27%)** bytes. These instances are referenced from one instance of **"java.lang.Object[]"**, loaded by **"<system class loader>"**

Keywords
java.lang.Object[]
com.enflick.android.TextNow.views.AvatarView
dalvik.system.PathClassLoader @ 0x32c02ce0

Details »

4. **PromoCampaignAd**
    a. Suspected due to large HashMap being stored inside class. HashMap object name is **sImageMap**
    b. Large Memory Use (5.97% of Heap Allocation) (May want to convert to SparseArray)

class com.enflick.android.TextNow.ads.PromoCampaignAd @ 0x338f8800 **System Clas**	120	7,506,080	5.97%
java.util.HashMap @ 0x33c6fe50	48	7,505,296	5.97%

Conclusion

There are **memory leaks** in our application. Java's Garbage Collector is not able to remove several objects due to strong references of them still existing in our application.

Suggest deeper analysis into the following classes in our Application

1. TintedImageView
2. AvatarImageView <Resolved>
3. Leanplum
4. PromoCampaignAd <Resolved>