DESCRIPTION dj...@gmail.com created issue #1

Target 21
Running on 19

Every once in a while, when a host doesn't exist (or doesn't respond) the `newCall( request ).execute();` will not spawn an `IOException` but the following error message, followed by a SIGFAULT.

According to square this seems to be an android issue: https://github.com/square/okhttp/issues/1192

It's uncatchable.

```
[ A/art : art/runtime/check_jni.cc:64] JNI DETECTED ERROR IN APPLICATION: JNI GetIntField called with pending exception 'java.net.ConnectException' thrown in boolean libcore.io.IoBridge.co
[ A/art : art/runtime/check_jni.cc:64]     in call to GetIntField
[ A/art : art/runtime/check_jni.cc:64]     from java.net.SocketAddress libcore.io.Posix.getsockname(java.io.FileDescriptor)
[ A/art : art/runtime/check_jni.cc:64] "IntentService[.CurrentUserApiService]" prio=5 tid=16 Runnable
[ A/art : art/runtime/check_jni.cc:64]   | group="main" sCount=0 dsCount=0 obj=0x64f22320 self=0x49c653e8
[ A/art : art/runtime/check_jni.cc:64]   | sysTid=22054 nice=0 cgrp=apps sched=0/0 handle=0x49a4b508
[ A/art : art/runtime/check_jni.cc:64]   | state=R schedstat=( 100017602 58703337 220 ) utm=10 stm=0 core=3 HZ=100
[ A/art : art/runtime/check_jni.cc:64]   | stack=0x4a0ec000-0x4a0f0000 stackSize=1040KB
[ A/art : art/runtime/check_jni.cc:64]   native: art::Thread::DumpStack(std::ostream&) const+87 [0x416a2c98] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: ??? [0x4154f97e] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: art::JniAbortF(char const*, char const*, ...)+51 [0x41550310] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: ??? [0x4155221c] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: ??? [0x41552486] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: ??? [0x41555cc0] (libart.so)
[ A/art : art/runtime/check_jni.cc:64]   native: jniGetFDFromFileDescriptor+77 [0x4025fdaa] (libnativehelper.so)
[ A/art : art/runtime/check_jni.cc:64]   native: ??? [0x42a5209c] (libjavacore.so)
[ A/art : art/runtime/check_jni.cc:64]   at libcore.io.Posix.getsockname(Native method)
[ A/art : art/runtime/check_jni.cc:64]   at libcore.io.ForwardingOs.getsockname(ForwardingOs.java:72)
[ A/art : art/runtime/check_jni.cc:64]   at libcore.io.IoBridge.getSocketLocalPort(IoBridge.java:592)
[ A/art : art/runtime/check_jni.cc:64]   at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:196)
[ A/art : art/runtime/check_jni.cc:64]   at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:461)
[ A/art : art/runtime/check_jni.cc:64]   at java.net.Socket.connect(Socket.java:834)
[ A/art : art/runtime/check_jni.cc:64]   - locked <0x64f5fad8> (a java.lang.Object)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.internal.Platform$Android.connectSocket(Platform.java:219)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.Connection.connect(Connection.java:152)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.Connection.connectAndSetOwner(Connection.java:171)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.OkHttpClient$1.connectAndSetOwner(OkHttpClient.java:120)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.internal.http.RouteSelector.next(RouteSelector.java:135)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:314)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:240)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.Call.getResponse(Call.java:235)
[ A/art : art/runtime/check_jni.cc:64]   at com.squareup.okhttp.Call.execute(Call.java:84)
[ A/art : art/runtime/check_jni.cc:64]   at .v3.EvergiveApi.execute(EvergiveApi.java:194)
[ A/art : art/runtime/check_jni.cc:64]   at .v3.service.ApiService.onHandleIntent(ApiService.java:191)
[ A/art : art/runtime/check_jni.cc:64]   - locked <0x64f1b888> (a java.lang.Object)
[ A/art : art/runtime/check_jni.cc:64]   at android.app.IntentService$ServiceHandler.handleMessage(IntentService.java:66)
[ A/art : art/runtime/check_jni.cc:64]   at android.os.Handler.dispatchMessage(Handler.java:102)
[ A/art : art/runtime/check_jni.cc:64]   at android.os.Looper.loop(Looper.java:138)
[ A/art : art/runtime/check_jni.cc:64]   at android.os.HandlerThread.run(HandlerThread.java:62)
[ A/art : art/runtime/check_jni.cc:64]
[ A/libc : Fatal signal 11 (SIGSEGV) at 0x2f64696f (code=1), thread 22054 (UserApiService])
```

✓ Links (6)

“According to square this seems to be an android issue: https://github.com/square/okhttp/issues/1192”

“ …ve transitioned back to managed code and back into native code (inside cacheLocalAddress) so the pending exception should've been thrown and caught by managed code in the interim. This look

“https://android-review.googlesource.com/#/c/117067 didn't spring to mind immediately: KitKat didn't have the offending change in. It also affects different methods in Posix, doesn't it? That problen

“https://code.google.com/p/android/issues/detail?id=94560&thanks=945… ”

“If the issue happens only when the debugger is attached, it is likely an issue with deoptimization. There was a fix in AOSP for this: https://android-review.googlesource.com/123720 .”

“… to address the issue reported, however our product team has shifted work priority that doesn't include this issue. For now, we will be closing the issue as won't fix obsolete. If this issue currently sti

COMMENTS

en...@google.com <en...@google.com>

Assigned to nf...@google.com.

---

nf...@google.com <nf...@google.com> #2

Status: New

Thank you for the report.

Initial analysis:

From the stack trace this is on KitKat with ART. It occurs somewhere inside jniGetFDFromFileDescriptor with an app-bundled OkHttp.

GetIntField is being called with a JNI exception (ConnectException from IoBridge.connect()) queued up. The stack trace printed seems to be from the failing call, and the queued one could ha
pending" checks are executed, but I assume fairly regularly.

I cannot reconcile the line numbers or callstack for PlainSocketImpl with the klp release code. There is no call to IoBridge.getSocketLocalPort() from PlainSocketImpl.connect().

I can't rule out that ART (which was experimental at this point) may be confusing the stack trace, or that this was an OEM-modified codebase. There *is* a call to getSocketLocalPort() in the

PlainSocketImpl thoughts / analysis / guesswork:

It's guesswork, but if I wanted to add code that happens after connect(), I would add it at the end of the method, which corresponds with line 196 (and has the closing brace for me). Line 461

The IoBridge.connect() call being reported as the originator of the exception has the correct signature as the call being made in PlainSocketImpl just above line 196.

IoBridge.connect() details:

Line 114 on klp-mr2-release is:

114: throw new ConnectException(connectDetail(inetAddress, port, timeoutMs, errnoException), errnoException);

This looks like a standard exception throw. No reason I can see why it wouldn't have popped out of PlainSocketImpl.connect(), since it extends IOException.

Conclusion pending more information:

ART was experimental in KitKat and turned off by default. Much has changed in it since. The simplest explanation is that this *may* be a bug in ART, since fixed.

The code is not (as far as I can see) what was released for KitKat by Google (any of kitkat-release, kitkat-mr1-release, kitkat-mr2-release), making repeating it or investigating further hard.

More information I would like before investigating further:

1) More details of the device / release installed on it if you have it. e.g. manufacturer, model, build number (and any other information you think may be useful!).

2) Whether this happens with Dalvik.

3) Whether this has been seen on other releases / devices.

---

en...@google.com <en...@google.com> #3

(the VM checks for pending exceptions any time native code calls back into the VM. so basically any time the native code says env->Something().)

---

da...@gmail.com <da...@gmail.com> #4

I believe I'm seeing the same on lollipop. target 21, running 21.

```
A/art : art/runtime/check_jni.cc:65] JNI DETECTED ERROR IN APPLICATION: JNI GetIntField called with pending exception 'java.net.SocketTimeoutException' thrown in void libcore.io.IoBrid
A/art : art/runtime/check_jni.cc:65]     in call to GetIntField
A/art : art/runtime/check_jni.cc:65]     from java.net.SocketAddress libcore.io.Posix.getsockname(java.io.FileDescriptor)
A/art : art/runtime/check_jni.cc:65] "AsyncTask #3" prio=5 tid=21 Runnable
A/art : art/runtime/check_jni.cc:65]   | group="main" sCount=0 dsCount=0 obj=0x12f920e0 self=0xaf94c400
A/art : art/runtime/check_jni.cc:65]   | sysTid=31601 nice=10 cgrp=apps/bg_non_interactive sched=0/0 handle=0xaf95c080
A/art : art/runtime/check_jni.cc:65]   | state=R schedstat=( 21171612 13365156 44 ) utm=1 stm=1 core=2 HZ=100
A/art : art/runtime/check_jni.cc:65]   | stack=0xa14fe000-0xa1500000 stackSize=1036KB
A/art : art/runtime/check_jni.cc:65]   | held mutexes= "mutator lock"(shared held)
A/art : art/runtime/check_jni.cc:65]   native: #00 pc 00004c58  /system/lib/libbacktrace_libc++.so (UnwindCurrent::Unwind(unsigned int, ucontext*)+23)
A/art : art/runtime/check_jni.cc:65]   native: #01 pc 000034c1  /system/lib/libbacktrace_libc++.so (Backtrace::Unwind(unsigned int, ucontext*)+8)
A/art : art/runtime/check_jni.cc:65]   native: #02 pc 002526c5  /system/lib/libart.so (art::DumpNativeStack(std::__1::basic_ostream<char, std::__1::char_traits<char> >&, int, char const*, art::
A/art : art/runtime/check_jni.cc:65]   native: #03 pc 002361a3  /system/lib/libart.so (art::Thread::Dump(std::__1::basic_ostream<char, std::__1::char_traits<char> >&) const+162)
A/art : art/runtime/check_jni.cc:65]   native: #04 pc 000b1215  /system/lib/libart.so (art::JniAbort(char const*, char const*)+620)
A/art : art/runtime/check_jni.cc:65]   native: #05 pc 000b1945  /system/lib/libart.so (art::JniAbortF(char const*, char const*, ...)+68)
A/art : art/runtime/check_jni.cc:65]   native: #06 pc 000b4bcd  /system/lib/libart.so (art::ScopedCheck::ScopedCheck(_JNIEnv*, int, char const*)+1324)
A/art : art/runtime/check_jni.cc:65]   native: #07 pc 000b8929  /system/lib/libart.so (art::CheckJNI::GetIntField(_JNIEnv*, _jobject*, _jfieldID*)+32)
A/art : art/runtime/check_jni.cc:65]   native: #08 pc 0001f65d  /system/lib/libjavacore.so (???)
A/art : art/runtime/check_jni.cc:65]   native: #09 pc 00270a8f  /data/dalvik-cache/arm/system@framework@boot.oat (Java_libcore_io_Posix_getsockname__Ljava_io_FileDescriptor_2+102
A/art : art/runtime/check_jni.cc:65]   at libcore.io.Posix.getsockname(Native method)
A/art : art/runtime/check_jni.cc:65]   at libcore.io.ForwardingOs.getsockname(ForwardingOs.java:87)
A/art : art/runtime/check_jni.cc:65]   at libcore.io.IoBridge.getSocketLocalAddress(IoBridge.java:629)
A/art : art/runtime/check_jni.cc:65]   at java.net.Socket.cacheLocalAddress(Socket.java:1001)
A/art : art/runtime/check_jni.cc:65]   at java.net.Socket.connect(Socket.java:884)
A/art : art/runtime/check_jni.cc:65]   - locked <0x11401c98> (a java.lang.Object)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.Platform.connectSocket(Platform.java:140)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.Connection.connect(Connection.java:150)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:277)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:215)
```

```
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.http.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:374)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.http.HttpURLConnectionImpl.connect(HttpURLConnectionImpl.java:106)
A/art : art/runtime/check_jni.cc:65]   at com.android.okhttp.internal.http.HttpURLConnectionImpl.getOutputStream(HttpURLConnectionImpl.java:210)
A/art : art/runtime/check_jni.cc:65]   at com.example.io.FetchJSON.Fetch(FetchJSON.java:45)
A/art : art/runtime/check_jni.cc:65]   at com.example.io.FetchJSON.Fetch(FetchJSON.java:21)
A/art : art/runtime/check_jni.cc:65]   at com.example.app.app.NearbyQueryTask$Fetcher.doFetch(NearbyQueryTask.java:40)
A/art : art/runtime/check_jni.cc:65]   at com.example.app.location.PlaceLookup.doInBackground(PlaceLookup.java:46)
A/art : art/runtime/check_jni.cc:65]   at com.example.app.location.PlaceLookup.doInBackground(PlaceLookup.java:23)
A/art : art/runtime/check_jni.cc:65]   at com.example.android.io.AsyncTaskBase.doInBackground(AsyncTaskBase.java:66)
A/art : art/runtime/check_jni.cc:65]   at android.os.AsyncTask$2.call(AsyncTask.java:288)
A/art : art/runtime/check_jni.cc:65]   at java.util.concurrent.FutureTask.run(FutureTask.java:237)
A/art : art/runtime/check_jni.cc:65]   at android.os.AsyncTask$SerialExecutor$1.run(AsyncTask.java:233)
A/art : art/runtime/check_jni.cc:65]   at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1120)
A/art : art/runtime/check_jni.cc:65]   at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:588)
A/art : art/runtime/check_jni.cc:65]   at java.lang.Thread.run(Thread.java:820)
A/art : art/runtime/check_jni.cc:65]
```

---

na...@google.com <na...@google.com> #5

the trace looks pretty weird. according to the error message and the code, the "pending" exception must have been thrown at line 882 at Socket.java.

```
882:        impl.connect(remoteAddr, timeout);
883:        isConnected = true;
884:        cacheLocalAddress();
```

We've transitioned back to managed code and back into native code (inside cacheLocalAddress) so the pending exception should've been thrown and caught by managed code in the interim

---

ag...@google.com <ag...@google.com> #6

A minimal reproducable example would be nice. Tweaking timeouts should make this rather fast-failing.

---

nf...@google.com <nf...@google.com> #7

https://android-review.googlesource.com/#/c/117067 didn't spring to mind immediately: KitKat didn't have the offending change in. It also affects different methods in Posix, doesn't it? That

It is a bit mysterious though, so I wouldn't entirely rule it out for L. Perhaps the necessary JNI exception state is corrupted and that exception *has* been thrown but the state hasn't been clea

Same pattern, similar locations of the "pending" exception, but different exceptions. Same getsockname() call, though a different part. That might be selection bias, though based on wha Dav

---

dj...@gmail.com <dj...@gmail.com> #8

The issue I had with earlier releases of ART on Kitkat (nexus-4, 4.4, .1 .2
and .4) was a JINI pin-error overflow pin-table and then all the threads
listed.

Your analysis is correct. Sorry for not providing that information; it
seemed clear from the Stacktrace.
Yes on the exception: Should have been thrown at the connect in the managed
code.

Mostly happens when the server is not responding (reachable), but not when
the internet connection itself is flaky. Takes a fair amount of time (>30
s). This seems to happen with Service-based implementations a lot where
timeouts take sometimes up to a minute to propagate (why?).

I can't recall if I had this on Dalvik; I only seem to get in with the
debugger attached.

Seen on:

One Plus
A0001
Android: 4.4.4 (KTU84Q)
CyanogenMod: 11.0-XNPh44S
Kernel: 3.4.0-cyanogenmod-gd8c0761

Nexus 4
Android: 4.4.4

I'll see if I can reproduce on 5.0.1 Nexus 4, 4.4.4 Nexus 7 / 5.0.1 Nexus 7.

Report back later this week.

---

da...@gmail.com <da...@gmail.com> #9

re: selection bias. I was searching for the top error line (art/runtime/check_jni.cc:65). Mine is also happening when my server is unreachable, with a timeout of 45 seconds.

---

nf...@google.com <nf...@google.com> #10

David, thanks for the info. Is this also when the debugger is attached, or do you see it at other times?

Regarding the server not being reachable: I'm wondering how best to repeat / simulate this. dj.gaymail@ suggested the server in his case was reachable, but not responding. david.watson@ taking place, but the server is under load so I assume the server socket accept() is taking place, but is then not handing off the resulting Socket for handling.

The OkHttp code in question makes several network connection attempts for one logical connection if it fails, depending on proxy settings, the number of IPs mapped to the host name, TLS I

david / dj: if you are not able to provide a repro case we can look at, if you can both send us some more contextual information we'd be grateful.

Starting with:
1) Nature of the connection timeout/failure
2) Connection configuration. Any proxy settings and details or nature of the connection (e.g. whether the host you're using maps to multiple IPs)
3) If this connection is over SSL or TLS, and any details you have there of the nature of that communication.

Any wireshark or similar logs or analysis you have done could also be useful.

---

dj...@gmail.com <dj...@gmail.com> #11

nful@

1) Only happens when the connection is remotely refused. Both host reachable (ie. server not runnning on port but resolves and target is online) but unresponsive as well as host unreachable
2) Both with direct ip address and hostname -> single ip address. No proxy. Both over WIFI and over the following: Edge, 3G, HSPA+ and LTE.
3) SSL for the hostname, no SSL for the direct ip address. Happens on all instances.

Havn't been able to reproduce it without the debugger, but from my memory it did crash without debugger, just took waaaay longer (like 1+ minutes).

Server load didn't have to do anything with it in these cases (as there was no backend running).

I'll be happy to provide anything more helpful. Just tell me (time constr. can't just turn the server off) exactly what you need.

---

nf...@google.com <nf...@google.com> #12

dj and/or David:

I need a reproduction case and I cannot repeat it based on what I currently understand with a lollipop build on a Nexus 5. If you could take the code I attach and tweak it to demonstrate the p

The problem may be device or OEM-Android-version specific, or there may be other factors at work (e.g. threading) so we should try to get to the simplest thing that demonstrates the probler

Perhaps you'll spot something in the code that indicates I don't properly understand the code or problem you have. Alternatively, if there is some kind of app code you are able to share I can l

I use the vogar tool to run this (as a standard test, and as an activity), and unfortunately it requires a rooted device. However, it could probably be adapted to run as a standard instrumented t

If we get a repro case we may have to start looking at tool like tcpdump so I can see the nature of the TCP traffic, and details of the build / device you're running.

First step though is to get something I can run that demonstrates the problem for you consistently and we are able to share.

---

deleted
0 B

---

km...@gmail.com <km...@gmail.com> #13

I am also facing "Network unreachable" problem when connecting socket in andoird 5.0 lollipop.
If i start that process from init.<device>.rc, then socket connection is successful. or if i set uid and gid as AID_ROOT, then its socket is connecting.

Kindly help me to solve the issue.

---

nf...@google.com <nf...@google.com> #14

kmsivambigai@ - please open a new bug. Nothing you've said suggests it is related to this issue. You'll need to provide much more detail to help somebody work out what the actual problem

---

bd...@google.com <bd...@google.com> #15

kmsivambigai, feel free to reference the new bug here so we can get it assigned appropriately. it seems more like an selinux or similar issue.

---

km...@gmail.com <km...@gmail.com> #16

@nful & @b..

Thanks for your reply. I raised separate new bug.
https://code.google.com/p/android/issues/detail?id=94560&thanks=94560&ts=1420781777

---

e....@gmail.com <e....@gmail.com> #17

I just encountered the "JNI DETECTED ERROR IN APPLICATION: JNI GetIntField called with pending exception 'java.net.SocketTimeoutException' thrown in void libcore.io.IoBridge.connectErr

I use a local web server (192.168.x.x) for development. Today I tried launching the app with the web server shut down to test if the app behaves correctly (which should have the exception su

The crash log looks quite similar to that of the first post, but I'm attaching it here anyway. My device is Nexus 5 GSM on Android 5.0.1 (LRX22C).

deleted
0 B

nf...@google.com <nf...@google.com> #18

Any chance of attaching code that demonstrates the problem?

e....@gmail.com <e....@gmail.com> #19

I encountered the same errors 2-3 times this afternoon, then left it alone after discovering that it's probably not an issue in my code and continued working. I turned back to this issue at night

nf...@google.com <nf...@google.com> #20

This is the problem with this bug. On paper it looks easily reproducible, but the nature of the stack trace suggests a problem in the runtime itself, or a threading issue, or both.

If we can get a repro case it should be easy to diagnose or direct to the right engineer. I've spent quite a lot of time trying to reproduce it and cannot, but I'm probably missing some kind of in

nf...@google.com <nf...@google.com> #21

If somebody does see this, can you report the following information:

1) If you had a debugger attached at the time.
2) Whether you had breakpoints set and what type. e.g. line Vs method Vs exception and whether those breakpoints were set to suspend one or all threads.

dj...@gmail.com <dj...@gmail.com> #22

#20 #21 @nful

Quick answer:
1. yes
2. line only. all threads. // line + method. all threads.

I am swamped with work but in february I will have some time to try to build a minimal case for you. Sorry I can't help more right now.

an...@gmail.com <an...@gmail.com> #23

@nful

I am using Android 5.0 and developing an app which communicates to my server. For testing i was running server on my local machine. Sometimes my server stops itself and in that case my

02-06 16:42:46.841 19989-20095/com.myPackage E/BaseLocationService : Network error
02-06 16:43:02.058 19989-20095/com.myPackage A/art : art/runtime/check_jni.cc:65] JNI DETECTED ERROR IN APPLICATION: JNI GetIntField called with pending exception 'java.net.Sock
02-06 16:43:02.058 19989-20095/com.myPackage A/art : art/runtime/check_jni.cc:65] in call to GetIntField

bd...@google.com <bd...@google.com> #24

agampe and I were discussing and he thinks this looks like it was a deopt problem, especially given the debugger usage. don't have a reference for what might have fixed it handy. adding she

sh...@google.com <sh...@google.com> #25

If the issue happens only when the debugger is attached, it is likely an issue with deoptimization. There was a fix in AOSP for this: https://android-review.googlesource.com/123720.

dj...@gmail.com <dj...@gmail.com> #26

Was?

Have been having a lot of crashes on ART + Debugger attached.

bd...@google.com <bd...@google.com> #27

"was" means it was in the past, on "Jan 23 [2015] 8:51 AM". That should be in the M developer preview release if you can try to reproduce there, it would be appreciated.

dj...@gmail.com <dj...@gmail.com> #28

@#27 Naturally. I will try to reproduce it there

sa...@google.com <sa...@google.com> #29

Status: Won't Fix (Obsolete)

Thank you for your feedback. We assure you that we are doing our best to address the issue reported, however our product team has shifted work priority that doesn't include this issue. For r