



Android Netfilter Changes

Linux Plumbers Conference 2013-09
jpa@google.com

Overview

- In The Beginning
- Requirements for Better Network Statistics
- How is **xt_qtaguid** used?
- How are **xt_quota2** + **xt_IDLETIMER** used?
- Features of **xt_qtaguid** vs using **xt_owner** + **xt_nfacct**

LPC: <http://www.linuxplumbersconf.org/2013/ocw/sessions/1491>

LKML: <http://lwn.net/Articles/517358/> by john.stultz@linaro.org

In The Beginning

- 2011-Q1
 - Needed a better data usage tracking mechanism
 - Had to work on 2.6.35 for Android Gingerbread
 - Nothing useful in the Kernel at that time

Requirements for Better Network Statistics

- Capture traffic per application and service
 - Handle services that act on behalf of other applications or services
 - A single connection might be used by different services over time
- Distinguish between downloads, video, chat, sync, ...
 - Current TrafficStats APIs are limited
- Prepare for user-based ~~traffic shaping~~
 - Support notion of quota (device, app, ...)
 - Limit traffic per application
 - Prevent one application from using another's bandwidth
- Move away from *hints* to application and into *enforcement*
- Allow for third-party applications to track and collect their own data

How is xt_qtaguid used?

overview

- **(Quota)**, socket **Tagging**, **UID** tracking
- Track all ingress/egress packets
- Track all interface statistics
- Let applications *tag* their own sockets
- Let applications *delegate* their own sockets
- Count SKB against looked-up Tag+UID
- Handle ipv4/ipv6
- Replaces `drivers/misc/uid_stats.c`, ...

<https://source.android.com/devices/tech/datausage/index.html>

<https://source.android.com/devices/tech/datausage/kernel-overview.html>

How are **xt_quota2** + **xt_IDLETIMER** used?

xt_quota2

- Limit data usage on certain classes of interfaces
- Original xtables-addons quota2 (named, SMP, shared)
- **uevent** on quota hit

xt_IDLETIMER

- Help ConnectivityService deal with quiet interfaces
- Original nf IDLETIMER
- uevent instead of sysfs
- Track when interface becomes **active** again

Features of xt_qtaguid

vs using xt_owner + xt_nfacct

- Avoid manipulating iptables during normal operations
 - Only needs basic accounting rules at startup
 - Apps can tag their communications via a simple procfs entry.
 - Simple Foreground vs background traffic accounting
- Avoid excessive iptables rules
 - Single rule for all installed app
 - Same single rule for all tagged communication of every app.
- Deal with interface stats of unknow interfaces
- Uses connection tracker to handle skb with no sk

Discussion

- + netfilter changes
- + Network Namespaces vs `ANDROID_PARANOID_NETWORK`
- + ...

BACKUP SLIDES

Focus on xt_qtaguid



Overview

- In The Beginning
- Requirements for Better Network Statistics
- What was available in 2011-Q1?
- What does xt_qtaguid actually do?
- Features of xt_qtaguid vs using xt_owner + xt_nfacct
- Netfilter support-changes in Android
- How xt_qtaguid fits into Android

LPC: <http://www.linuxplumbersconf.org/2013/ocw/sessions/1491>

LKML: <http://lwn.net/Articles/517358/> by john.stultz@linaro.org

In The Beginning

- 2011-Q1
 - Needed a better data usage tracking mechanism
 - Had to work on 2.6.35 for Android Gingerbread
 - Nothing useful in the Kernel at that time

Requirements for Better Network Statistics

- Capture traffic per application and service
 - Handle services that act on behalf of other applications or services
 - A single connection might be used by different services over time
- Distinguish between downloads, video, chat, sync, ...
 - Current TrafficStats APIs are limited
- Prepare for user-based ~~traffic shaping~~
 - Support notion of quota (device, app, ...)
 - Limit traffic per application
 - Prevent one application from using another's bandwidth
- Move away from *hints* to application and into *enforcement*
- Allow for third-party applications to track and collect their own data

What was available in 2011-Q1?

investigation

- Traffic Control and QoS
- Aliased NICs and bridges
- Expanding cgroups for NICs
- Network Namespaces
- Hacking Netfilter Modules (NFACCT is 2011-Q4)

Custom Netfilter module with custom tags seemed the way to go

What does xt_qtaguid actually do?

overview

- **(Quota)**, socket **T**agging, **U**ID tracking
- Track all ingress/egress packets
- Track all interface statistics
- Let applications *tag* their own sockets
- Let applications *delegate* their own sockets
- Count SKB against looked-up Tag+UID
- Handle ipv4/ipv6
- Replaces `drivers/misc/uid_stats.c`, ...

<https://source.android.com/devices/tech/datausage/index.html>

<https://source.android.com/devices/tech/datausage/kernel-overview.html>

What does xt_qtaguid actually do?

Track all ingress/egress packets

- Track all ingress/egress packets

- system/netd/BandwidthController.cpp

```
const char *BandwidthController::IPT_BASIC_ACCOUNTING_COMMANDS[] = {  
    "-A bw_INPUT -m owner --socket-exists", /* This is a tracking rule. */  
    "-A bw_OUTPUT -m owner --socket-exists", /* This is a tracking rule. */  
    "-t raw -A bw_raw_PREROUTING -m owner --socket-exists", /* This is a tracking rule. */  
    "-t mangle -A bw_mangle_POSTROUTING -m owner --socket-exists", /* This is a tracking rule. */  
};
```

- Data available via TrafficStats API

- Natively via `/proc/net/xt_qtaguid/stats`

What does xt_qtaguid actually do?

Track interface stats

- Track interface stats

- net/netfilter/xt_qtaguid.c `qtaguid_mt()`

```
switch (par->hooknum) {  
    case NF_INET_PRE_ROUTING:  
    case NF_INET_POST_ROUTING:  
        atomic64_inc(&qtu_events.match_calls_prepost);  
        iface_stat_update_from_skb(skb, par);  
        /*  
         * We are done in pre/post. The skb will get processed  
         * further alter.  
         */  
        res = (info->match ^ info->invert);  
        goto ret_res;  
        break;  
    /* default: Fall through and do UID related work */  
}
```

- Tracks any new interface
- More accurate than `/proc/net/dev`

What does `xt_qtaguid` actually do?

Let applications *tag* their own sockets

- Let applications to *tag* their own sockets
 - Custom tags allow tracking different sections of a IP stream (E.g. setup, authentication, data0, data1, ...)
 - Via `/proc/net/xt_qtaguid/ctrl`
 - No need for any extra privileges or talk to any framework API
 - Works for native apps

What does xt_qtaguid actually do?

Let applications *delegate* their own sockets

- Let applications *delegate* their own sockets
 - Privileged applications can set tag+UID on socket they own
 - Data accounting against other tag+UIDs
 - Happy download managers and sync services

What does `xt_qtaguid` actually do?

Assign SKB to Tag+UID

- Assign SKB to Tag+UID
 - Orphaned SKBs go to `root`
 - SKBs with un-tagged sockets default to the UID owning the socket.
 - SKBs with tagged socket go to the UID in the Tag+UID of the socket

Features of xt_qtaguid

vs using xt_owner + xt_nfacct

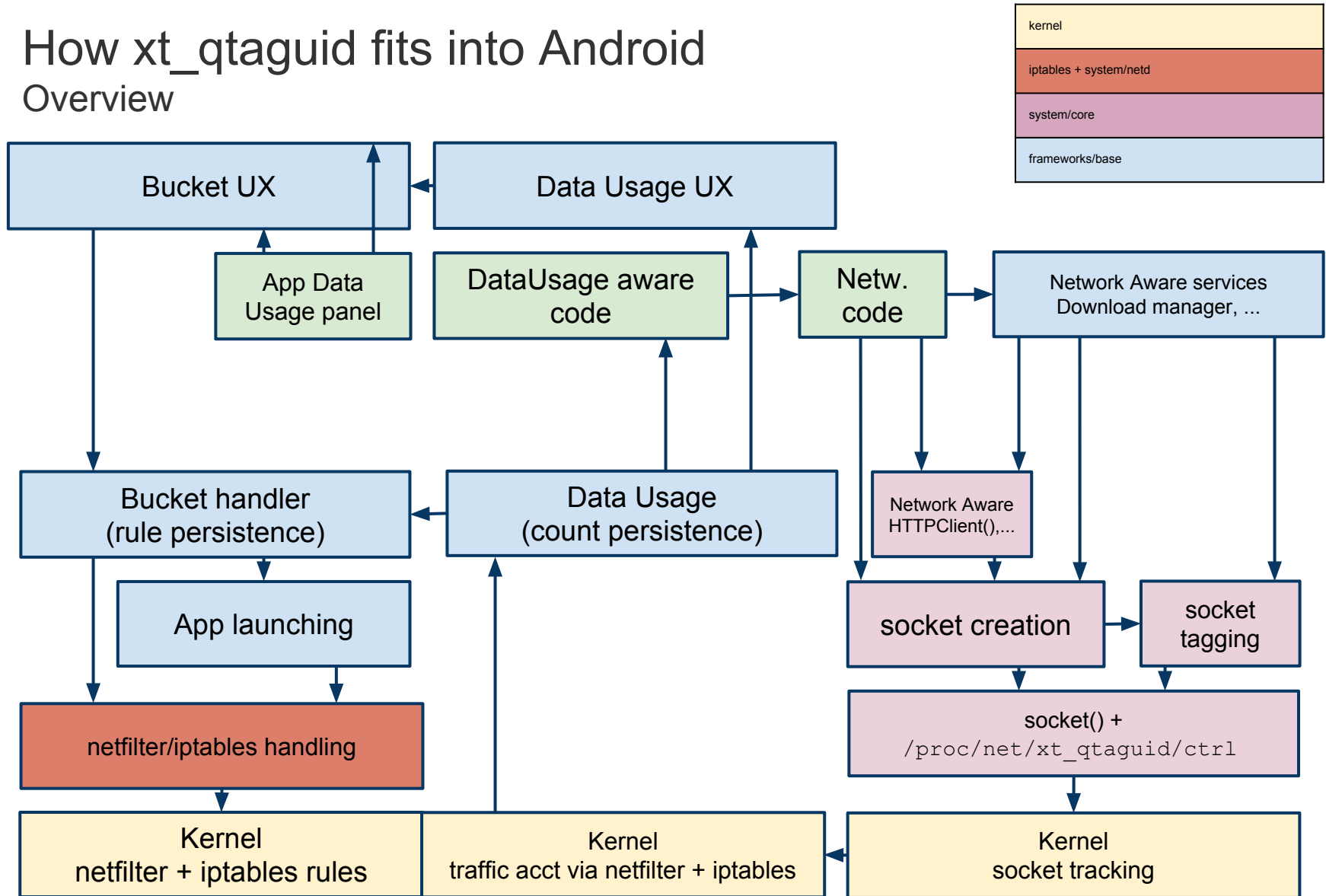
- Avoid manipulating iptables during normal operations
 - Only needs basic accounting rules at startup
 - Apps can tag their communications via a simple procfs entry.
 - Simple Foreground vs background traffic accounting
- Avoid excessive iptables rules
 - Single rule for all installed app
 - Same single rule for all tagged communication of every app.
- Deal with interface stats of unknow interfaces
- Uses connection tracker to handle skb with no sk

Netfilter support-changes in Android

- kernel
 - tweaked xt_socket, new xt_qtaguid, imported xt_quota2
- external/iptables
 - fixes, import xt_quota2 from xtables-addons
- system/netd
 - BandwidthController, iptables sharing, ...
- system/core/libcutils
- frameworks/base/core
 - jni, NetworkManagementSocketTagger, ...
- CTS tests against socket tagging

How xt_qtaguid fits into Android

Overview



How xt_qtaguid fits into Android

Action: System startup

- setup default global rules
- scan installed apps
- load custom app rules
- load prior data usage, update counters/adjust leftovers

How xt_qtaguid fits into Android

Action: Dalvik starts up app

- get app A specific rules
- ~~setup app's policing rules using policing_apptagA for all~~

How xt_qtaguid fits into Android

Action: App uses sockets

- opens socket (classic)
- sends data
- traffic controller kicks in
 - netfilter module sees skb
 - no policing_*tag for socket, assigns **policing_apptagX=f(app_UID)**
 - nf module counts data against tags
 - ~~policing tags picked up by traffic controller rules, shaping applied (dropped if limit reached)~~

How xt_qtaguid fits into Android

Action: Data Usage aware App uses sockets

- Open socket
- Requests sock_fd to be marked with custom acct_apptagA
- Send data
- Traffic controller kicks in
 - netfilter module sees skb
 - no policing_*tag for socket, assigns policing_apptagA
 - skbuf tracked with acct_apptagA
 - nf module counts data against matching tags (... , apptagA)
 - policing_tag picked up by traffic controller rules, shaping applied (drop if limit reached)
- App marks socket with acct_tagB in qtaguid module
- Send more data
- Traffic controller kicks in
 - ... skbuf tracked with acct_apptagB ...

How xt_qtaguid fits into Android

Action: Download manager ++

- Get request from app A (uid)
- Get tags for uid
 - v0: policing
 - ~~later: accounting~~
- open socket
- mark socket with policing_apptagA in qtaguid module
- (only dl manager is allowed to set policing tags)
- send data
- traffic controller kicks in
 - nf modules sees policing tag already set
 - nf module counts data against matching tags (... , apptagA)
 - ~~tc policing tag picked up by traffic controller rules, shaping applied (drop if limit reached)~~

What next?

Q&A