



MEDIATEK

Android Multi-network Framework

Doc No: AD6000-F3A-SHD-V1.0EN

Version: V1.0

Release date: 2016-08-28

Classification: Internal

© 2008 - 2017 MediaTek Inc.

This document contains information that is proprietary to MediaTek Inc.

Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

Specifications are subject to change without notice.



Error! No text of specified style in document.
Error! No text of specified style in document.

Keywords

Multi
network,ConnectivityService,Connectivity
Manager, System Design Document

MediaTek Inc.

Postal address

No. 1, Dusing 1st Rd. , Hsinchu Science
Park, Hsinchu City, Taiwan 30078

MTK support office address

No. 1, Dusing 1st Rd. , Hsinchu Science
Park, Hsinchu City, Taiwan 30078

Internet

<http://www.mediatek.com/>



Error! No text of specified style in document.

Error! No text of specified style in document.

Document Revision History

Document Revision History

Revision	Date	Author	Description
A1	2016-09-08	Roger Chang	Initial Release

MediaTek Confidential

© 2016 - 2017 MediaTek Inc.

Classification: Internal

Table of Contents

Document Revision History.....	3
Table of Contents.....	4
Lists of Tables	6
Lists of Figures	7
1 Introduction	8
1.1 Purpose	8
1.2 Scope	8
1.3 Who Should Read This Document	9
1.4 How to Use This Manual	9
1.4.1 Terms and Conventions	9
2 References.....	11
3 Definitions.....	12
4 Abbreviations.....	13
5 Multinetwork Introduction.....	14
5.1 Overall Architecture	14
5.1.1 NetworkFactory	15
5.1.2 NetworkAgent	18
5.1.3 NetworkCapability	20
5.2 The Scores	22
5.3 Working flow introduction	23
5.3.1 Create a Default Data Connection(Default network)	23
5.3.2 Default Network Switching	25
5.3.3 Make Wi-Fi & Mobile co-existence	25
6 Introduction to Linux Routing.....	27
6.1 Routing table Format	27
6.2 Routing Selection	28
6.2.1 Fwmark	29
6.2.2 Fwmark routing – IP rules	29
6.2.3 Fwmark Routing – Routing tables	30

Table of Contents

7	ConnectivityManager SDK API	32
7.1	Deprecated API (API Level: 21. Android 5.0 (LOLLIPOP))	32
8	Application development using Multinetwork SDK API	34
8.1	Create a network.....	34
8.2	Release a network.....	35
8.3	Network CallBack	35
8.4	Choose a Specific Network.....	36
8.4.1	Method-1 (Preferred).....	37
8.4.2	Method-2	37
9	System Requirements Traceability Matrix	38



Error! No text of specified style in document.
Error! No text of specified style in document.

Lists of Tables

Lists of Tables

Table 1-1. Reference Information beyond Scope..... 9

Table 1-2. Chapter Overview 9

Table 1-3. Conventions 9

Table 4-1. Abbreviations 13

Table 8-1. System Requirements Traceability Matrix 1-to-1 Table..... 38

Table 8-2. System Requirements Traceability Matrix 1-to-many Table..... 38



Error! No text of specified style in document.

Error! No text of specified style in document.

Lists of Figures

Lists of Figures

Figure 10-1. Title-Cased Caption Text. Centered.Error! Bookmark not defined.

Figure 10-2a. Sleepy. b. Tired.....Error! Bookmark not defined.

Figure 10-3. Chipmunks and Flowers and AlligatorsError! Bookmark not defined.

MediaTek Confidential

© 2016 - 2017 MediaTek Inc.

Classification: Internal

This document contains information that is proprietary to MediaTek Inc.
Unauthorized reproduction or disclosure of this information in whole or in part is strictly prohibited.

1 Introduction

☞ [Random filler text. Not intended for actual reading.] Must keep the chapter even it have empty content.

1.1 Purpose

This document provides the high level description for Android new design in connectivity named “Multi-network framework”. Android introduce the new framework since Lollipop trying to resolve complicated application and new technology functionality in the future(Wi-Fi&Mobile co-existence). It introduce new modules like NetworkFactory, NetworkAgent...etc. In the older version of Android, there is only one network can be active in a time e.g. Wi-Fi/Mobile. Therefor if enable Wi-Fi and Mobile data simutaneuosly, the Wi-Fi would be gevin higher priority, all data traffic would go through wlan interface.

This documentation not only explain framework design(new modules working follow) but also provide API usage for application developer who intent to use network functionality.

[Random filler text. Not intended for actual reading.] This section should describe about the following items:

- What does this document provide?
- What should the reader get after reading the document?
- Any concrete outcome (gains, or applications) can get after step-by-step following the document?

1.2 Scope

Android platform version, Lollipop, Marshmallow, Nougat

No chipset dependency

[Random filler text. Not intended for actual reading.] This section should describe about the following items:

- What hardware and version is this document applying? The hardware can be MTK chipset or external module. Please don't put multiple MTK chipset information here. If this document can apply multiple MTK chipset, please duplicates this document for each chipset and modify the chipset information.
- What platform and version is this document applying? For example, Android platform version and kernel version.
- What hardware and software module is used but beyond the scope? Please describe them and add reference after the module.

Table 1-1 presents the reference information of the modules which are used but beyond the scope.

Table 1-1. Reference Information beyond Scope

Modules	Reference information
NA	NA

1.3 Who Should Read This Document

This document is primarily intended for:

- Engineers with technical knowledge of the Connectivity framework
- Customers who using the ConnectivityManager SDK API with user-defined applications

1.4 How to Use This Manual

This segment explains how information is distributed in this document, and presents some cues and examples to simplify finding and understanding information in this document. Table 1-2 presents an overview of the chapters and appendices in this document.

Table 1-2. Chapter Overview

#	Chapter	Contents
1	Introduction	Describes the scope and layout of this document.
2	References	Reference info or link
3	Definitions	Term definitions
4	Abbreviations	
5	Multinetwork Introduction	Provide over view of multinetwork and major components description
6	Introduction to Linux Routing	Describes basic routing and advance routing for the relation with multiple network
7	ConnectivityManager SDK API	Quick view of the new SDK API for application impact and deprecated API review
8	Application development using Multinetwork SDK API	How to use new API to create/release network and how to specific a network to transfer the application packets

1.4.1 Terms and Conventions

This document uses special terms and typographical conventions to help you easily identify various information types in this document. These cues are designed to simply finding and understanding the information this document contains.

Table 1-3. Conventions


Convention	Usage	Example
------------	-------	---------



Error! No text of specified style in document.

Error! No text of specified style in document.

1 Introduction

Convention	Usage	Example
[1]	Serial number of a document in the order of appearance in the References topic	Look up Chapter 2: System Architecture in [1]
void xx(zz)	Source code	static int __stdcall cb_download_bloader_init(void *usr_arg){}
	Important	

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- [1] ConnectivityManager,
<https://developer.android.com/reference/android/net/ConnectivityManager.html>
- [2] 2015 Android bootcamp slides
- [3] Linux Routing
<http://linux-ip.net/html/routing-intro.html>

☞ [Random filler text. Not intended for actual reading.] Must keep the chapter even it have empty content.

3 Definitions

For the purposes of the present document, the following terms and definitions apply:

Default network: a network contains Internet capability

☞ [Random filler text. Not intended for actual reading.] Must keep the chapter even it have empty content.

4 Abbreviations

Please note the abbreviations and their explanations provided in Table 4-1. They are used in many fundamental definitions and explanations in this document and are specific to the information that this document contains.

Table 4-1. Abbreviations

Abbreviations	Explanation
MTK	MediaTek, Asia's largest fabless IC design company.
CS	Connectivity Service
NF	Network Factory
NA	Network Agent
FWK	Framework
SS	Signal Strength
PBR	Policy based routing

☞ [Random filler text. Not intended for actual reading.] Must keep the chapter even it have empty content.

5 Multinetwork Introduction

5.1 Overall Architecture

Multinetwork Architecture

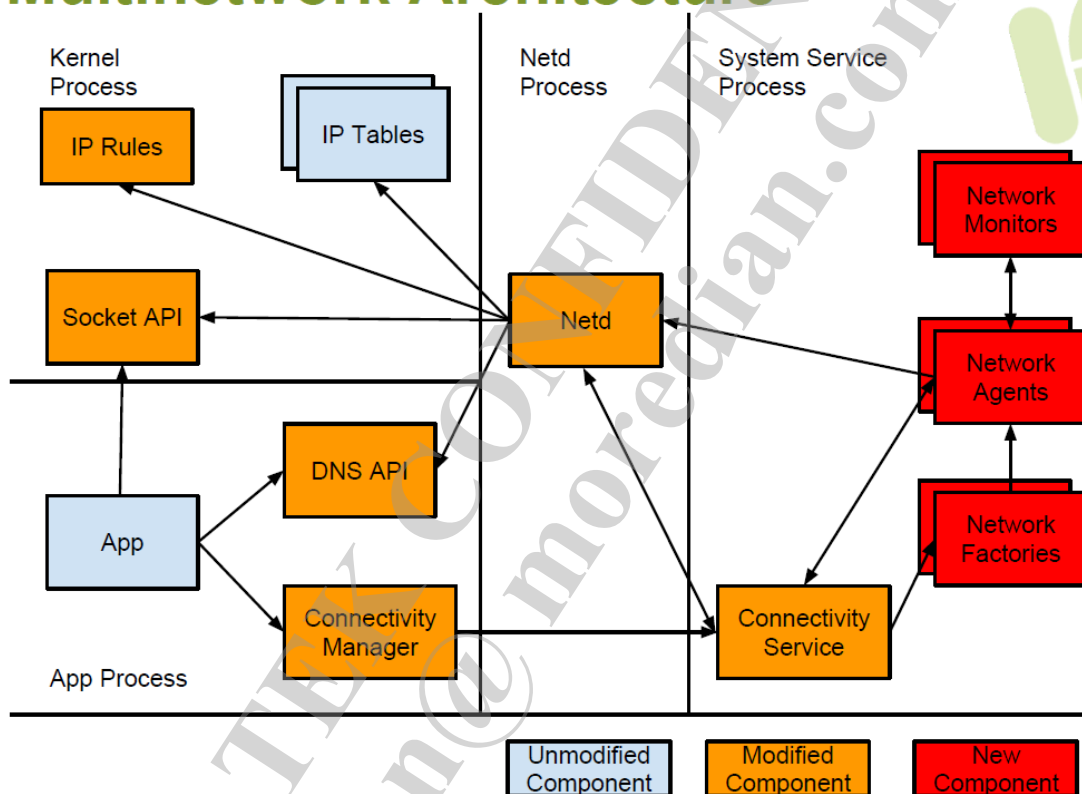


Figure 1. Overall Architecture

There are some new module are introduced in multinetwork architecture, networkFactory/networkAgent/NetworkMonitor...etc. The ConnectivityManager is modified for new SDK API design and ConnectivityService has big change for new desing.

Basically the new design idea is,

Control path: app/service request a network, networkFactory accept it and create the pdn connection and register networkAgent if connection is established successfully. Each IP rule define what routing table should be used.

Data path: Giving each packet a mark via socket option, which decides what routing table should be use.

5 Multinetwork Introduction

As illustrated in Figure 2, the networkFactory is living in each connectivity module which is provide the Connection capability(e.g. Wi-Fi/Mobile). networkAgents are registered in ConnectivityService for updating networkInfo like connecting/disconnected.

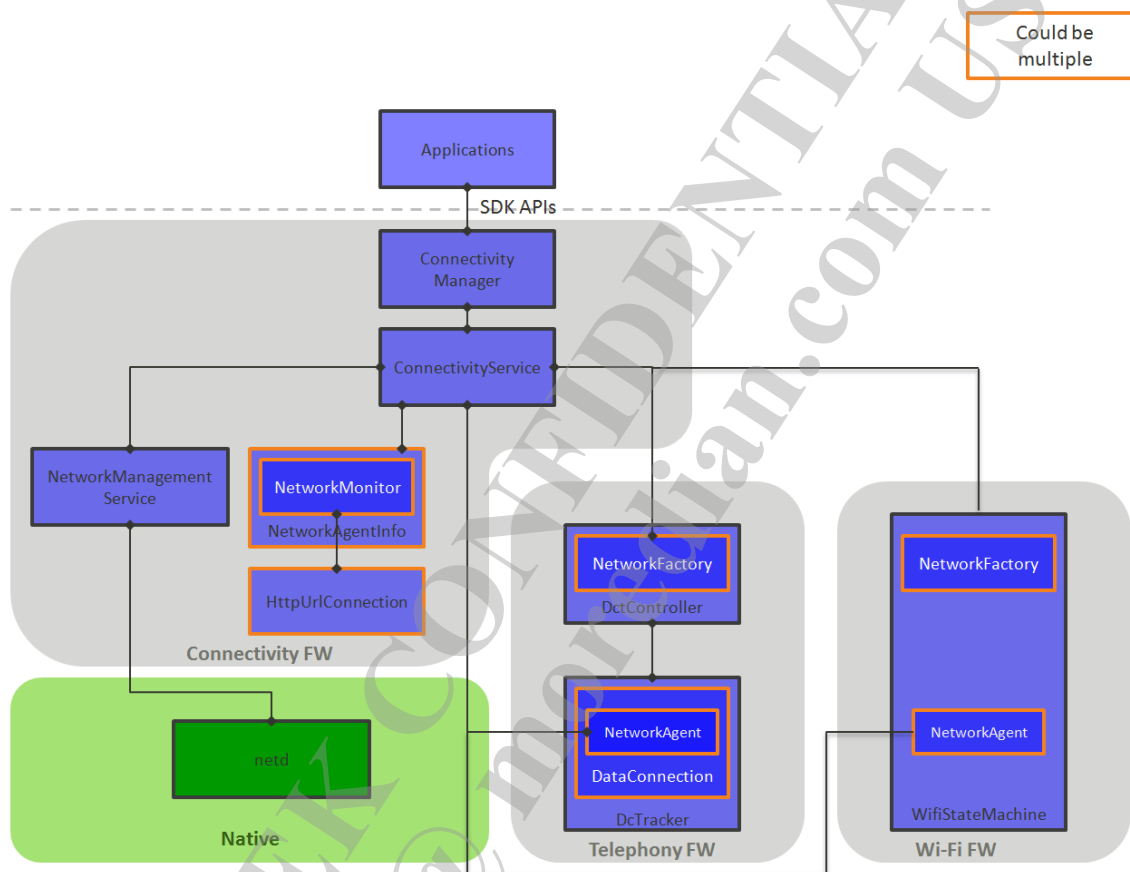


Figure 2. Multinetwork block diagram

[Random filler text. Not intended for actual reading.]

Use UML package diagrams and/or layer diagrams and/or interface diagrams and/or system block diagrams and/or context diagrams to illustrate the top level software components and their interactions/relationships.

5.1.1 NetworkFactory

NetworkFactory is extends from the Handler. The properties are as the following:

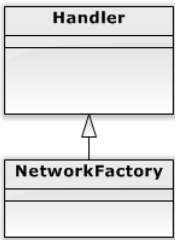


Figure 3. NetworkFactory Inherit

- A factory handles commands from ConnectivityService
- Register to ConnectivityService while system initing
- Communicate with *NetworkStateTrackerHandler* in ConnectivityService
- Async channel
 - One direction only



Figure 4. NetworkFactory Async channel direction

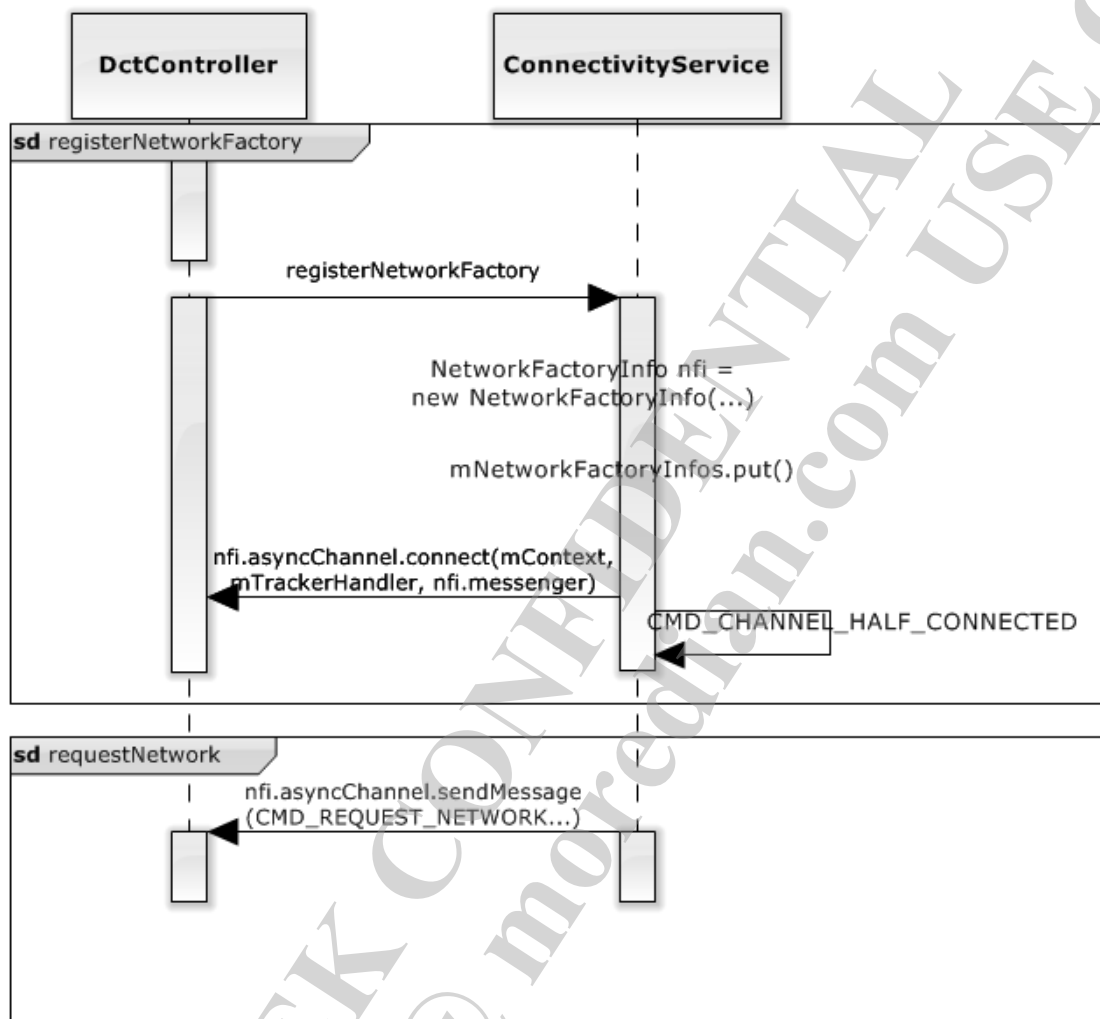


Figure 5. NetworkFactory Async channel creation

There is a command table, AOSP design 4 commands are issued from CS so far.

Commands	Description	Direction
CMD_REQUEST_NETWORK	Giving a network request to the network factory	CS -> NF
CMD_CANCEL_REQUEST	Releasing a network request to the network factory	CS -> NF

5 Multinetwork Introduction

CMD_SET_SCORE	Change the score* to the network factory	CS -> NF
CMD_SET_FILTER	Change the capability to the network factory	CS -> NF

Table 1. NetworkFactory command

[Random filler text. Not intended for actual reading.]

Describe the content of each top level software component in the architecture. The description shall contains:

- Its identification,
- The purpose of the component,
- Its interfaces with other components,
- The resources it uses or depends on, for example : RAM usage, frequency and duration, disk space for permanent data, disk space for cache data, CPU usage, and peak frequency and duration ...

5.1.2 NetworkAgent

NetworkAgent is extends from the Handler. The properties are as the following:

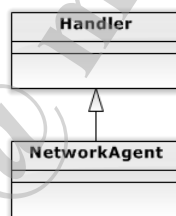


Figure 6. NetworkAgent Inherit

- An agent which update events to ConnectivityService
- Register to ConnectivityService while connection is creating(Wi-Fi) or created(Mobile)
- Communicate with *NetworkStateTrackerHandler* in ConnectivityService
- Async channel
 - Bi-direction

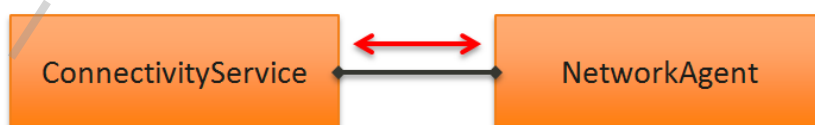


Figure 7. NetworkAgent Async channel direction

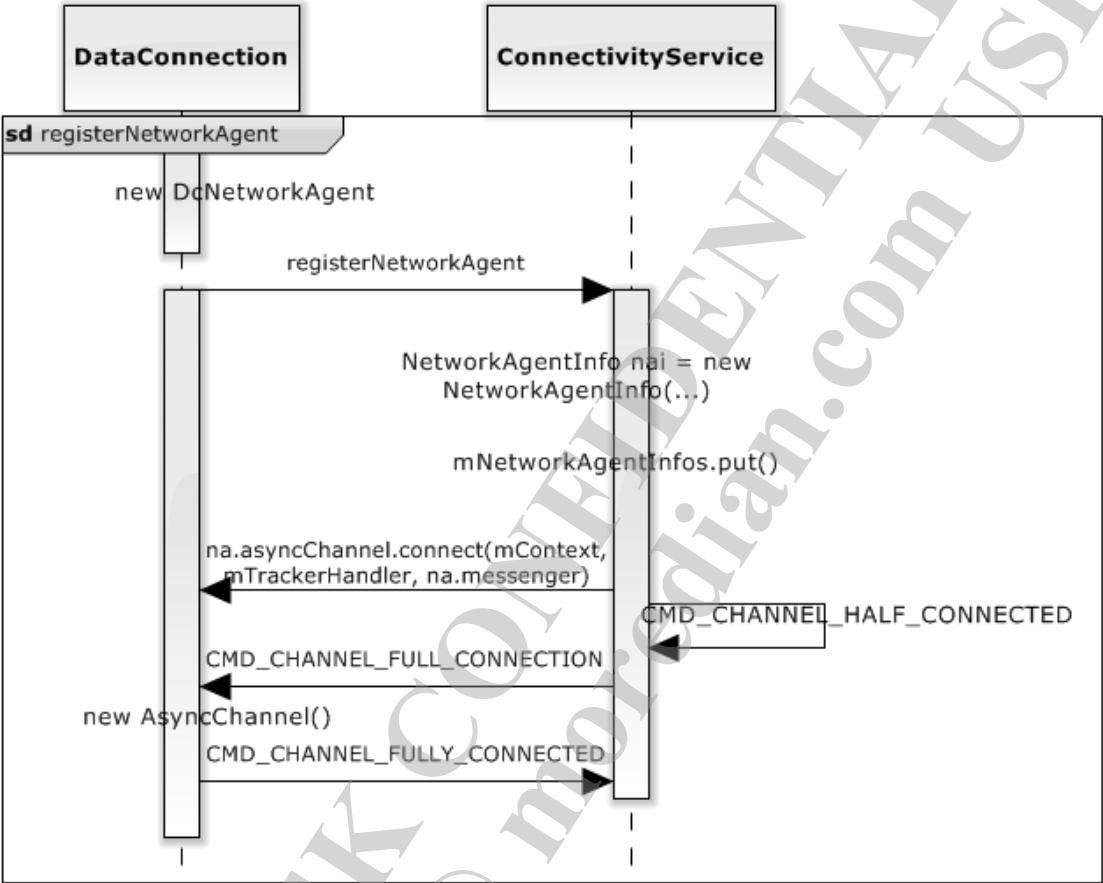


Figure 8. NetworkAgent Async channel creation

Commands/Events	Description	Direction
CMD_SUSPECT_BAD*	to inform it of suspected connectivity problems on its network. The NetworkAgent should take steps to verify and correct connectivity.	CS -> NA
EVENT_NETWORK_INFO_CHANGED	Sent when the NetworkInfo changes, mainly due to change of state	NA -> CS



Error! No text of specified style in document.

Error! No text of specified style in document.

5 Multinetwork Introduction

EVENT_NETWORK_CAPABILITIES_CHANGED	Sent when the NetworkCapabilities changes	NA -> CS
EVENT_NETWORK_PROPERTIES_CHANGED	Sent when the NetworkPropertieschanges	NA -> CS
EVENT_NETWORK_SCORE_CHANGED	to pass the current network score	NA -> CS
EVENT_UID_RANGES_ADDED	to add new UID ranges to be forced into this Network. For VPNs only.	NA -> CS
EVENT_UID_RANGES_REMOVED	to remove new UID ranges to be forced into this Network. For VPNs only.	NA -> CS
CMD_REPORT_NETWORK_STATUS	to inform the agent of the networks status - whether we could use the network or could not, due to either a bad network configuration (no internet link) or captive portal.	CS -> NA
EVENT_SET_EXPLICITLY_SELECTED	to indicate this network was explicitly selected. This should be sent before the NetworkInfo is marked CONNECTED so it can be given special treatment at that time.	NA -> CS

Table 2. NetworkAgent command/Event

5.1.3 NetworkCapability

Each module would be assigned a capability

- NetworkFactory
- NetworkAgent
- NetworkRequest

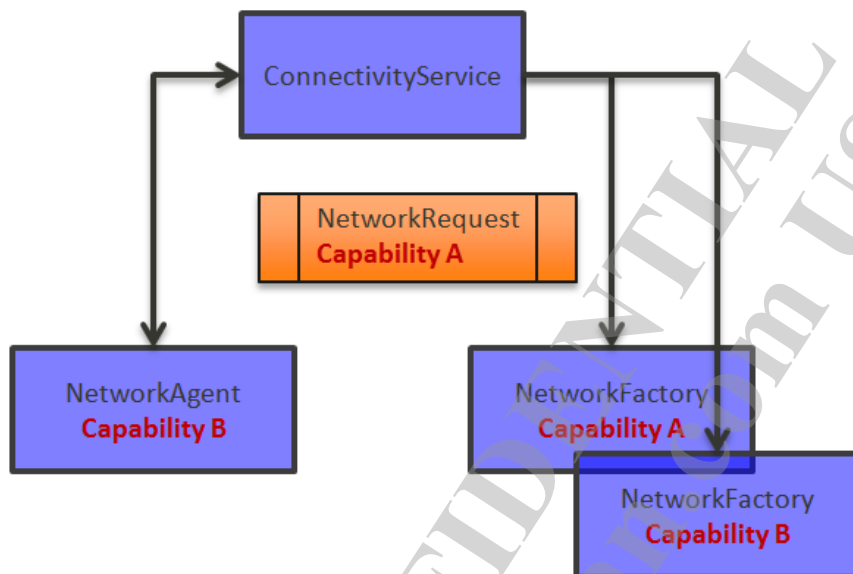


Figure 9. The capabilities of networkAgent and networkFactory

networkFacotories contains different capability as showed in Figure 9. In most case, NetworkAgent capability is a subset of the corresponding NetworkFactory and the socre is same from the networkFactory. But the score of networkAgent is changeable as previously mentioned. (e.g. Wi-Fi networkAgent change the score by signal strength, purpose is expect CS to choose mobile if SS is too bad)

Transport Types	Values
TRANSPORT_CELLULAR	0
TRANSPORT_WIFI	1
TRANSPORT_BLUETOOTH	2
TRANSPORT_ETHERNET	3
TRANSPORT_VPN	4

Table 3. Transport Types of networkCapability

5 Multinetwork Introduction

Capabilities	Values
NET_CAPABILITY_MMS	0
NET_CAPABILITY_SUPL	1
NET_CAPABILITY_DUN	2
NET_CAPABILITY_FOTA	3
NET_CAPABILITY_IMS	4
NET_CAPABILITY_CBS	5
NET_CAPABILITY_WIFI_P2P	6
NET_CAPABILITY_IA	7
NET_CAPABILITY_RCS	8
NET_CAPABILITY_XCAP	9
NET_CAPABILITY_EIMS	10
NET_CAPABILITY_NOT_METERED	11
NET_CAPABILITY_INTERNET	12
NET_CAPABILITY_NOT_RESTRICTED*	13
NET_CAPABILITY_TRUSTED*	14
NET_CAPABILITY_NOT_VPN*	15
NET_CAPABILITY_VALIDATED	16

Table 4. Capabilities of networkCapability

5.2 The Scores

The score is significant design in multinetwork framework.

- Decide which network would be dropped if same capability networks are exists.
- Decide NetworkRequest is acceptable by NetworkFactory

NetworkFactory Score

- TelephonyNetworkFactory fixed to 50
- WifiNetworkFactory fixed to 60
- EthernetNetworkFactory fixed to 70 (will not discuss here)
- Design to changeable, but all are fixed in L design

NetworkAgent Score

- TelephonyNetworkFactory initial is 50
- WifiNetworkAgent initial is 60
- Changeable(e.g. downgrade to 30 due to Wi-Fi RSSI is not good)

NetworkRequest Score

- Is assigned by ConnectivityService by different conditions
 - e.g.
 - ◆ bestNetwork's score
 - ◆ Zero if there is no existed network satisfied
- Changeable

[Random filler text. Not intended for actual reading.]

Describe. Use UML package diagrams and/or layer diagrams and/or interface diagrams and/or system block diagrams and/or context diagrams and/or components diagrams and/or deployment diagrams and/or network diagrams, to illustrate the hardware components on which software runs and their interactions/relationships.

5.3 Working flow introduction

In this section, we illustrated 3 significant scenarios to understand the interaction in each module.

(Including the score assigning, changing)

5.3.1 Create a Default Data Connection(Default network)

In this scenario, device is boot up with available SIM card inserted, user turn ON mobile data.

5 Multinetwork Introduction

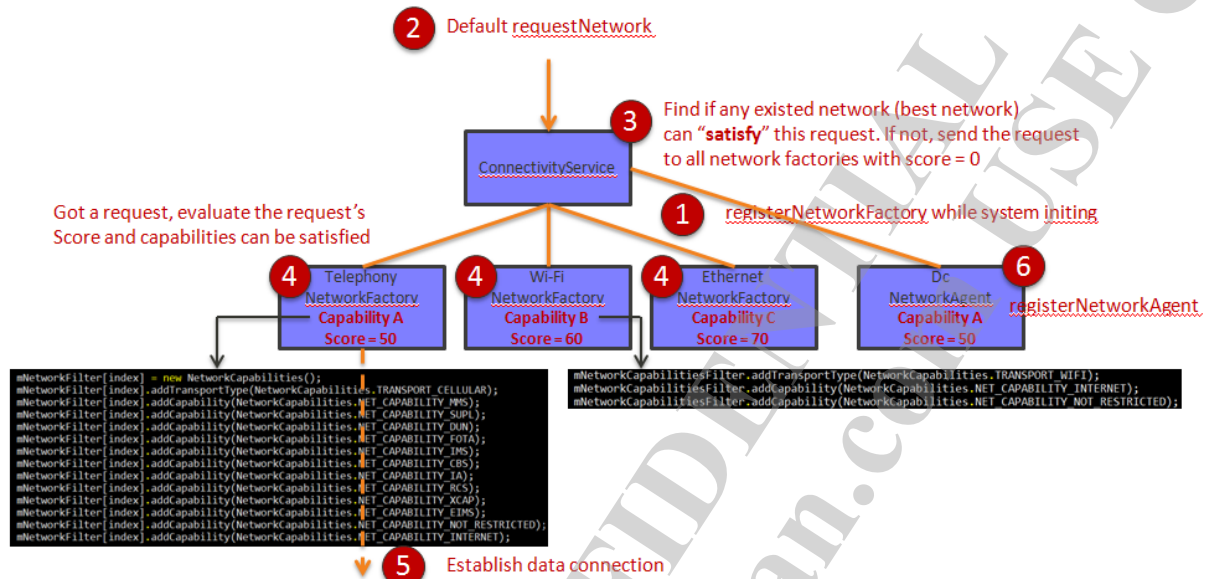


Figure 10. Create default network

Step 1. registerNetworkFactory while system initing

Step 2. Default networkRequest is issued from CS

Step 3. CS inspect if any existed network can satisfy the request. (Checking the score and NetworkCapabilities)
If no any network, CS assign the score to 0 in networkRequest.

Step 4. CS broadcasts the networkRequest to all networkFactories, TelephonyNetworkFactory score = 50, means it can satisfy the new networkRequest = 0. (if networkRequest score < networkFactory score)

Step 5. TelephonyNetworkFactory start to establish the data connection

Step 6. Telephony data connection is connected, register networkAgent to CS and update the networkRequest score to 50.

[Random filler text. Not intended for actual reading.]

Describe the content of each top level hardware component in the architecture. The description shall contains:

- Its identification,
- The purpose of the component,
- Its interfaces with other components,
- The software component it receives
- Its technical characteristics

5 Multinetwork Introduction

5.3.2 Default Network Switching

In this scenario, mobile data is connected, user turn ON Wi-Fi and successfully associated an AP(with Internet accessible).

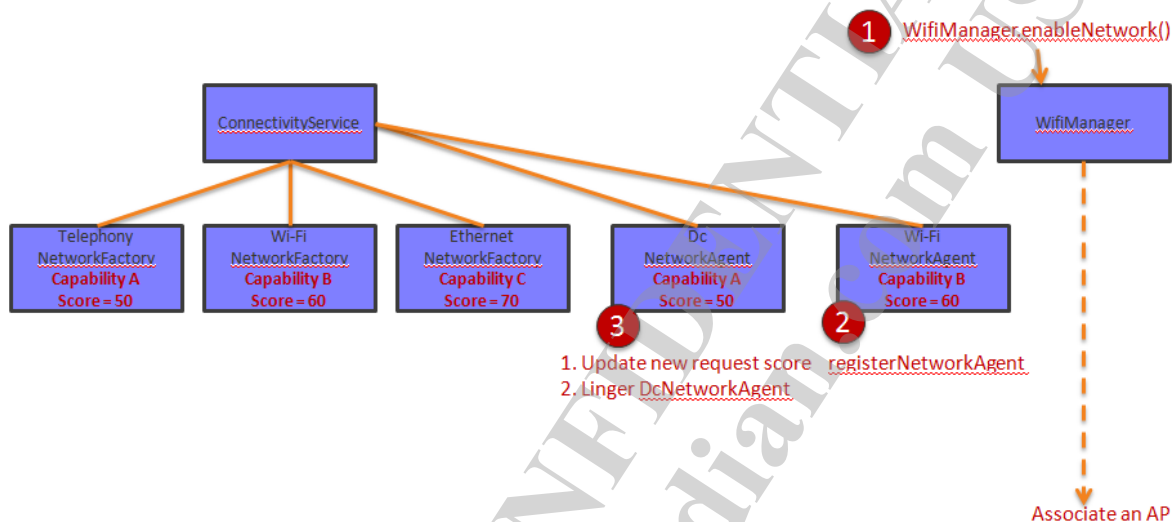


Figure 11. Default network switching

Step 0. Telephony networkAgent is registered to CS score = 50

Step 1. User select an Wi-Fi AP to connect and connected

Step 2. Wi-Fi networkAgent is registered to CS score = 60

Step 3. CS rematch networkRequest and networkAgent, there is only one networkRequest but two networkAgent(Wi-Fi, Mobile) are satisfied. CS compare the score of networkAgent, keep the higher score of networkAgent. Linger DcNetworkAgent. (mobile data would be disconnected)

5.3.3 Make Wi-Fi & Mobile co-existence

To support completed application usage, AOSP enhance a feature named “Moble Data Always On”

5 Multinetwork Introduction

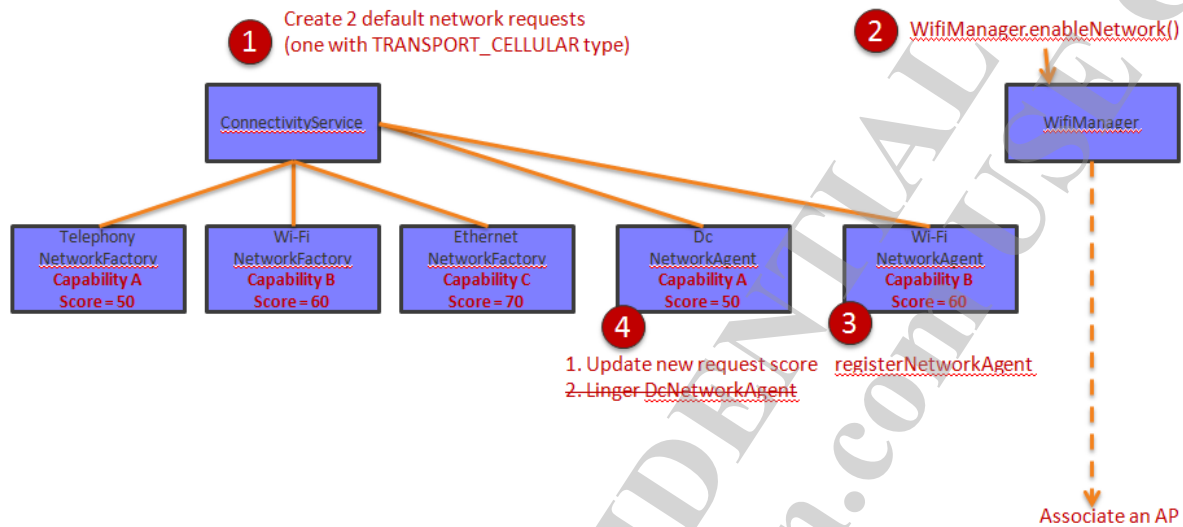


Figure 12. Wi-Fi & Mobile co-existence

Step 0. Telephony networkAgent is registered to CS score = 50

Step 1. Create secondary default networkRequest(MUST set transport type to cellular)

Step 2. User select an Wi-Fi AP to connect and connected

Step 3. CS rematch networkRequest and networkAgent,

Wi-Fi networkAgent satisfy default networkRequest 1

Telephony networkAgent satisfy default networkRequest 2(transport type to cellular)

CS decide no need to drop any networkAgent due to networkAgent all matched.

6 Introduction to Linux Routing

A key concept in IP routing is the ability to define what addresses are locally reachable as opposed to not directly known destinations

Three classes of destination:

- Itself
 - Loopback interface & local host address
- locally connected computers
 - IP addresses are addresses in the locally connected network segment
- Everywhere
 - All other hosts or destination IPs



Figure 13. Routing concept

6.1 Routing table Format

Linux routing table contains rules to mapping which packet should be applied and make the routing decision.

6 Introduction to Linux Routing

Destination	Network Mask	Gateway	Interface	Metric
192.168.1.0	255.255.255.0	192.168.0.254	wlan0	111

Figure 14. Routing table format

Basic Routing

The regular routing are 2 types

- Destination routing
- Source routing

```
ip route show
default via 172.23.14.254 dev wlan0 metric 312
172.23.14.0/24 dev wlan0 proto static
172.23.14.0/24 dev wlan0 proto kernel scope link src 172.23.14.19 metric 312
```

Figure 15. An Example of routing info in Android

Android use destination routing in most case.

6.2 Routing Selection

Except basic routing, the another powerful routing is introduced for complicated scenarios - PBR

1. Basic Routing (Destination)



Figure 16. Basic destination routing

2. Advanced Routing: Policy based routing (PBR)

Routing Selection Algorithm in Pseudo-code

```
if packet.routeCacheLookupKey in routeCache :
    route = routeCache[ packet.routeCacheLookupKey ]
else
    for rule in rpdb :
        if packet.rpdbLookupKey in rule :
            routeTable = rule[ lookupTable ]
            if packet.routeLookupKey in routeTable :
                route = route_table[ packet.routeLookup_key ]
```

Figure 16. Policy based routing pseudo-code

From the pseudo-code, the packet routing is decided by PBR algorithm

1. The routing table is selected by IP rule
2. The packet routing is by routing rule

Multinetwork uses this concept to make the packet routing.

[Random filler text. Not intended for actual reading.]

Each component should have a unique identifier. The identifiers to be used for components should be defined by the project and described elsewhere.

6.2.1 Fwmark

- **SO_MARK socket option**
 - Set the mark for each packet sent through this socket (similar to the netfilter MARK target but socket-based).
 - Changing the mark can be used for mark-based routing without netfilter or for packet filtering.
 - Setting this option requires the CAP_NET_ADMIN capability.
- **Example**
 - `setsockopt(client_socket, SOL_SOCKET, SO_MARK, &mark, sizeof(mark));`

6.2.2 Fwmark routing – IP rules

This is an example to explain the Fwmark routing in Android

No network connected:

6 Introduction to Linux Routing

```
0:      from all lookup local
10000:  from all fwmark 0xc0000/0xd0000 lookup legacy_system
13000:  from all fwmark 0x10063/0x1ffff lookup local_network
15000:  from all fwmark 0x0/0x10000 lookup legacy_system
16000:  from all fwmark 0x0/0x10000 lookup legacy_network
17000:  from all fwmark 0x0/0x10000 lookup local_network
23000:  from all fwmark 0x0/0xffff uidrange 0-0 lookup main
32000:  from all unreachable
```

Figure 17. IP routes

Wi-Fi network connected:

Wi-Fi network connected:

```
0:      from all lookup local
10000:  from all fwmark 0xc0000/0xd0000 lookup legacy_system
13000:  from all fwmark 0x10063/0x1ffff lookup local_network
13000:  from all fwmark 0x10064/0x1ffff lookup wlan0
14000:  from all oif wlan0 lookup wlan0
15000:  from all fwmark 0x0/0x10000 lookup legacy_system
16000:  from all fwmark 0x0/0x10000 lookup legacy_network
17000:  from all fwmark 0x0/0x10000 lookup local_network
19000:  from all fwmark 0x64/0x1ffff lookup wlan0
22000:  from all fwmark 0x0/0xffff lookup wlan0
23000:  from all fwmark 0x0/0xffff uidrange 0-0 lookup main
32000:  from all unreachable
```

```
const uint32_t RULE_PRIORITY_VPN_OVERRIDE_SYSTEM = 10000;
const uint32_t RULE_PRIORITY_VPN_OUTPUT_TO_LOCAL = 11000;
const uint32_t RULE_PRIORITY_SECURE_VPN          = 12000;
const uint32_t RULE_PRIORITY_EXPLICIT_NETWORK    = 13000;
const uint32_t RULE_PRIORITY_OUTPUT_INTERFACE   = 14000;
const uint32_t RULE_PRIORITY_LEGACY_SYSTEM       = 15000;
const uint32_t RULE_PRIORITY_LEGACY_NETWORK     = 16000;
const uint32_t RULE_PRIORITY_LOCAL_NETWORK      = 17000;
const uint32_t RULE_PRIORITY_TETHERING          = 18000;
const uint32_t RULE_PRIORITY_IMPLICIT_NETWORK   = 19000;
const uint32_t RULE_PRIORITY_BYPASSABLE_VPN     = 20000;
const uint32_t RULE_PRIORITY_VPN_FALLTHROUGH   = 21000;
const uint32_t RULE_PRIORITY_DEFAULT_NETWORK    = 22000;
const uint32_t RULE_PRIORITY_DIRECTLY_CONNECTED = 23000;
const uint32_t RULE_PRIORITY_UNREACHABLE        = 32000;
```

```
struct {
    unsigned netId      : 16;
    bool explicitlySelected : 1;
    bool protectedFromVpn : 1;
    Permission permission : 2;
}
```

Figure 18. IP rules – network connected

The IP rule 22000 is default network

Each IP rule priority is define by different purpose, in Android, if a packet no match 1000~19000 rules, it goes default network ip rule, and use wlan0 routing table in this example.

[Random filler text. Not intended for actual reading.]

- This section should describe the type of component, e.g. task, subroutine, subprogram, package, file.
- The contents of some component description sections depend on the component type. For the purpose of this template the categories: executable, i.e. contains computer instructions, or non-executable, i.e. contains only data, are used.

6.2.3 Fwmark Routing – Routing tables

After routing table is select, the packet is route by routing table, in this example wlan0 routing is used.

And the default route is wi-fi gateway 192.168.0.1.

No network connected:

Local table

6 Introduction to Linux Routing

```
broadcast 127.0.0.0 dev lo table local proto kernel scope link src 127.0.0.1
local 127.0.0.0/8 dev lo table local proto kernel scope host src 127.0.0.1
local 127.0.0.1 dev lo table local proto kernel scope host src 127.0.0.1
broadcast 127.255.255.255 dev lo table local proto kernel scope link src 127.0.0.1
```

Figure 19. Routing table

Wi-Fi network connected:

120.119.28.1 via 192.168.0.1 dev wlan0 table legacy_system proto static	legacy_system table
218.75.4.130 via 192.168.0.1 dev wlan0 table legacy_system proto static	legacy_system table
default via 192.168.0.1 dev wlan0 table wlan0 proto static	wlan0 table
192.168.0.0/24 dev wlan0 table wlan0 proto static scope link	wlan0 table
default via 192.168.0.1 dev wlan0 metric 310	
192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.106	main table
192.168.0.0/24 dev wlan0 proto kernel scope link src 192.168.0.106 metric 310	main table
broadcast 127.0.0.0 dev lo table local proto kernel scope link src 127.0.0.1	local table
local 127.0.0.0/8 dev lo table local proto kernel scope host src 127.0.0.1	local table
local 127.0.0.1 dev lo table local proto kernel scope host src 127.0.0.1	local table
broadcast 127.255.255.255 dev lo table local proto kernel scope link src 127.0.0.1	local table
broadcast 192.168.0.0 dev wlan0 table local proto kernel scope link src 192.168.0.106	
local 192.168.0.106 dev wlan0 table local proto kernel scope host src 192.168.0.106	
broadcast 192.168.0.255 dev wlan0 table local proto kernel scope link src 192.168.0.106	

Figure 20. Routing table – network connected

7 ConnectivityManager SDK API

7.1 Deprecated API (API Level: 21. Android 5.0 (LOLLIPOP))

Deprecated	Instead on L	Reason
EXTRA_NETWORK_INFO	getActiveNetworkInfo()/getAllNetworkInfo()	NetworkCallbacks are instead of Intents
public void setNetworkPreference (int preference)	NA	Functionality has been removed as it no longer makes sense, with many more than two networks
public int startUsingNetworkFeature (int networkType, String feature)	public void requestNetwork (NetworkRequest request, PendingIntent operation)	Multi-Network design
public int stopUsingNetworkFeature (int networkType, String feature)	public void unregisterNetworkCallback (NetworkCallback networkCallback)	Multi-Network design
public boolean requestRouteToHost (int networkType, int hostAddress)	requestNetwork(...) + setProcessDefaultNetwork(...) / getSocketFactory(...)	Multi-Network design
public boolean getMobileDataEnabled()	public void setDataEnabled (boolean enable) in TelephonyManager	NA
public static boolean setProcessDefaultNetworkForHostResolution (Network network)	This is a new API but deprecated	This is strictly for legacy usage to support startUsingNetworkFeature



Error! No text of specified style in document.

Error! No text of specified style in document.

7 ConnectivityManager SDK API

Table 5. ConnectivityManager deprecated API

For more latest update, please refer to Android developer website.

8 Application development using Multinetwork SDK API

8.1 Create a network

Setp 1. New networkRequest

```
import android.net.NetworkRequest;

private final NetworkRequest mNetworkRequest;

mNetworkRequest = new NetworkRequest.Builder()

    .addTransportType(NetworkCapabilities.TRANSPORT_CELLULAR)

    .addCapability(NetworkCapabilities.NET_CAPABILITY_MMS)

    .setNetworkSpecifier(Integer.toString(mSubId))

    .build();
```

Step 2. Implement your callback function

```
import android.net.ConnectivityManager;

mNetworkCallback = new ConnectivityManager.NetworkCallback() {

    @Override

    public void onAvailable(Network network) {

        super.onAvailable(network);

        Log.d(TAG, "NetworkCallbackListener.onAvailable: network=" + network);

        //TODO:

    }

    @Override

    public void onLost(Network network) {

        super.onLost(network);

        Log.d(TAG, "NetworkCallbackListener.onLost: network=" + network);

        //TODO:

    }

    @Override
```

```
public void onUnavailable() {
    super.onUnavailable();

    //TODO:
}
};
```

Step 3. Send networkRequest to ConnectivityManager

```
ConnectivityManager.requestNetwork(mNetworkRequest, mNetworkCallback);
```

8.2 Release a network

```
connectivityManager.unregisterNetworkCallback(mNetworkCallback);
```

How to get the result of network releasing(disconnected)??

Answer: Currently not support to get release result

8.3 Network Callback

10 types are defined but only 7 types are available for Apps

Call back API Names	Description	Is in using
onPreCheck(Network network)	Called whenever the framework connects to a network that it may use to satisfy this request (Network is connected + not validated)	Yes
onAvailable(Network network)	Called when the framework connects and has declared new network ready for use. This callback may be called more than once if the Network that is satisfying the request changes. (Network is connected + no matter validated/not validated)	Yes
onLosing(Network network, int maxMsToLive)	Called when the network is about to be	Yes

8 Application development using Multinetwork SDK API

	disconnected. Often paired with an NetworkCallback#onAvailable call with the new replacement network for graceful handover. This may not be called if we have a hard loss (loss without warning). This may be followed by either a NetworkCallback#onLost call or a link NetworkCallback#onAvailable call for this network depending on whether we lose or regain it. (maxMsToLive now is hard code to 30 sec)	
onLost(Network network)	Called when the framework has a hard loss of the network or when the graceful failure ends. (called ONLY if disconnected by network)	Yes
onUnavailable()	Called if no network is found in the given timeout time. If no timeout is given, this will not be called. (skip it)	No (except MTK ePDG feature)
onCapabilitiesChanged(Network network, NetworkCapabilities networkCapabilities)	Called when the network the framework connected to for this request changes capabilities but still satisfies the stated need.	Yes
onLinkPropertiesChanged(Network network, LinkProperties linkProperties)	Called when the network the framework connected to for this request changes LinkProperties.	Yes

Table 6. Network callback

8.4 Choose a Specific Network

If multiple networks are connected, e.g. mms pdn, internet pdn and Wi-Fi are connected. How an application choose a network corresponding their purpose?

8.4.1 Method-1 (Preferred)

1. Get the Network by *onAvailable(Network network)*
2. *Network.getSocketFactory().createSocket()*
 - Any Socket created by this factory will have its traffic sent over this Network
3. *Network.getAllByName(String host)*
 - Operates the same as "*InetAddress.getAllByName*" except that host resolution is done on this network

8.4.2 Method-2

1. Get the Network by *onAvailable(Network network)*
2. *ConnectivityManager.setProcessDefaultNetwork(Network network)*
 - Binds the current process to the *network*. All Sockets created in the future will be bound to the *network*.
 - All host name resolutions will be limited to the *network* as well.
 - *ConnectivityManager.setProcessDefaultNetwork(null)* - Clear the binding
3. *ConnectivityManager.setProcessDefaultNetworkForHostResolution(Network network)*
 - *setProcessDefaultNetwork* takes precedence over this setting



Error! No text of specified style in document.

Error! No text of specified style in document.

9 System Requirements Traceability Matrix

9 System Requirements Traceability Matrix

Table 9-1. System Requirements Traceability Matrix 1-to-1 Table

System Req. Number	System Ref. Item	Component Identifier	Component Item

Table 9-2. System Requirements Traceability Matrix 1-to-many Table

System Req. Number	System Ref. Item	Component Identifier	Component Item