通过一个网络请求完美封装一个网络模块

Allen

2019.10

20:10课程正式开始

大纲和往期视频请加阿缘老师QQ：2807762965

享学讲师团队

Alvin老师

曾就职于三星、小米，项目经理

Leo 老师

某创业公司技术总监，网易特约讲师

King 老师

曾就职于招行、58同城

Allen老师

国防科大研究生毕业，全球首批Android开发者

Zero老师

前阿里P7移动架构师，曾就职于Nubia等一线互联网公司。

Lance老师

某游戏公司主程，前爱奇艺高程。

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

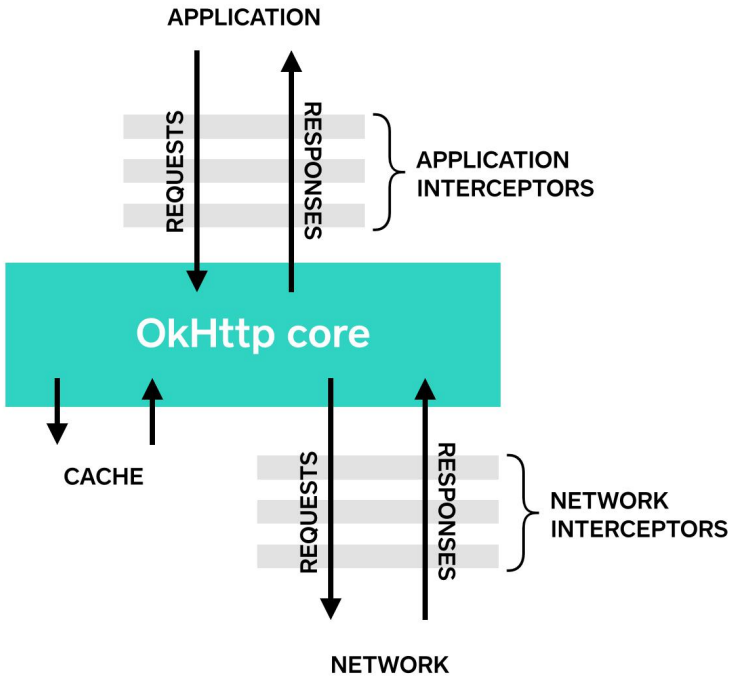
5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

1.1、当今占统治地位的网络组件



1.2、App架构设计

业务模块

common

网络

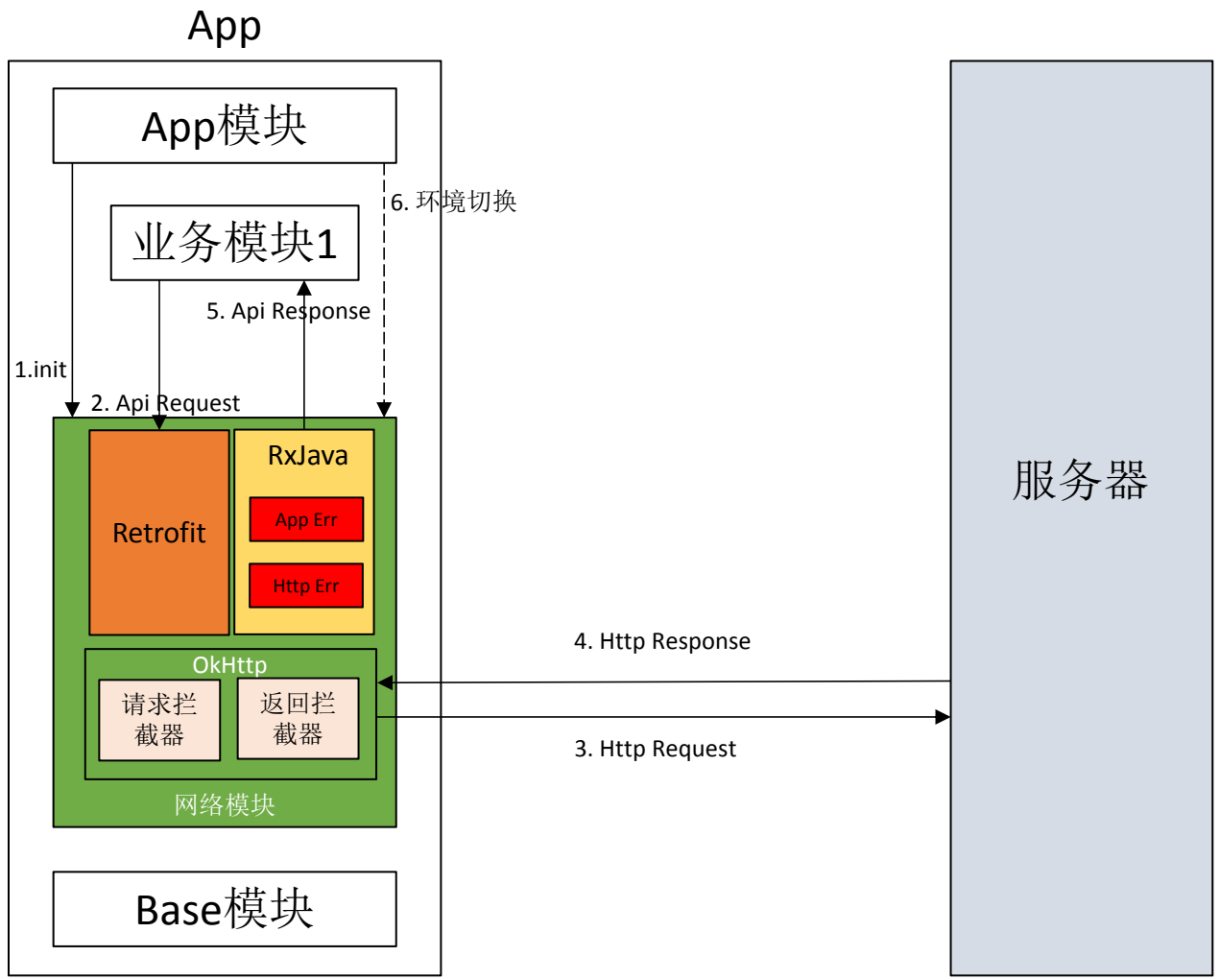
Base

JNI库

Android操作系统

硬件

1.3、网络层架构


App
App模块
业务模块1
6. 环境切换
5. Api Response
1.init
2. Api Request
Retrofit
RxJava
App Err
Http Err
OkHttp
请求拦截器
返回拦截器
网络模块
Base模块
4. Http Response
3. Http Request
服务器

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

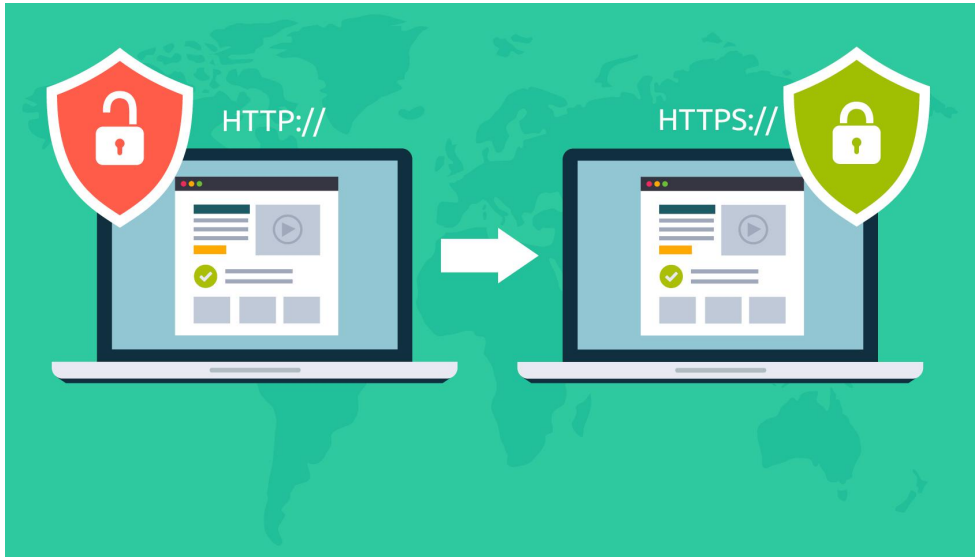
5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

2.1、Http/Https

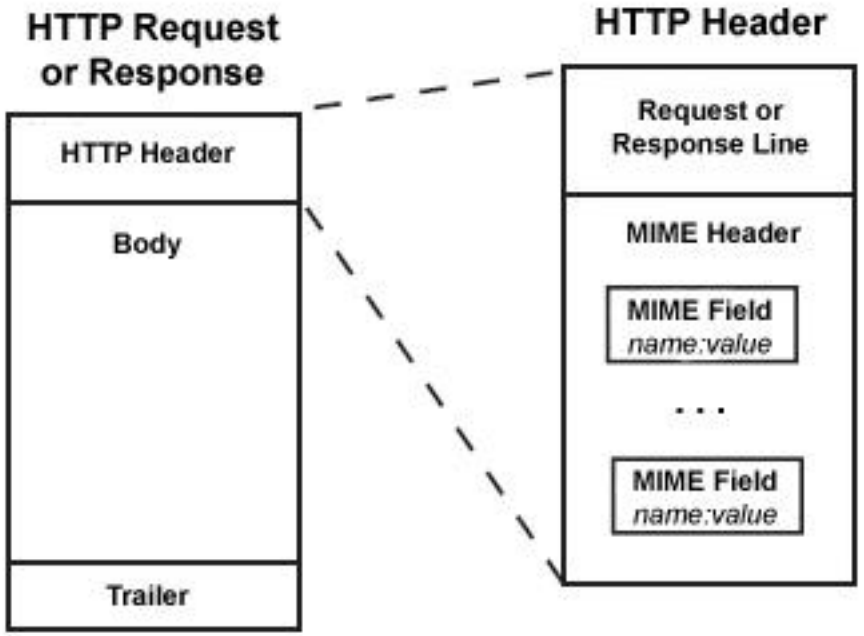


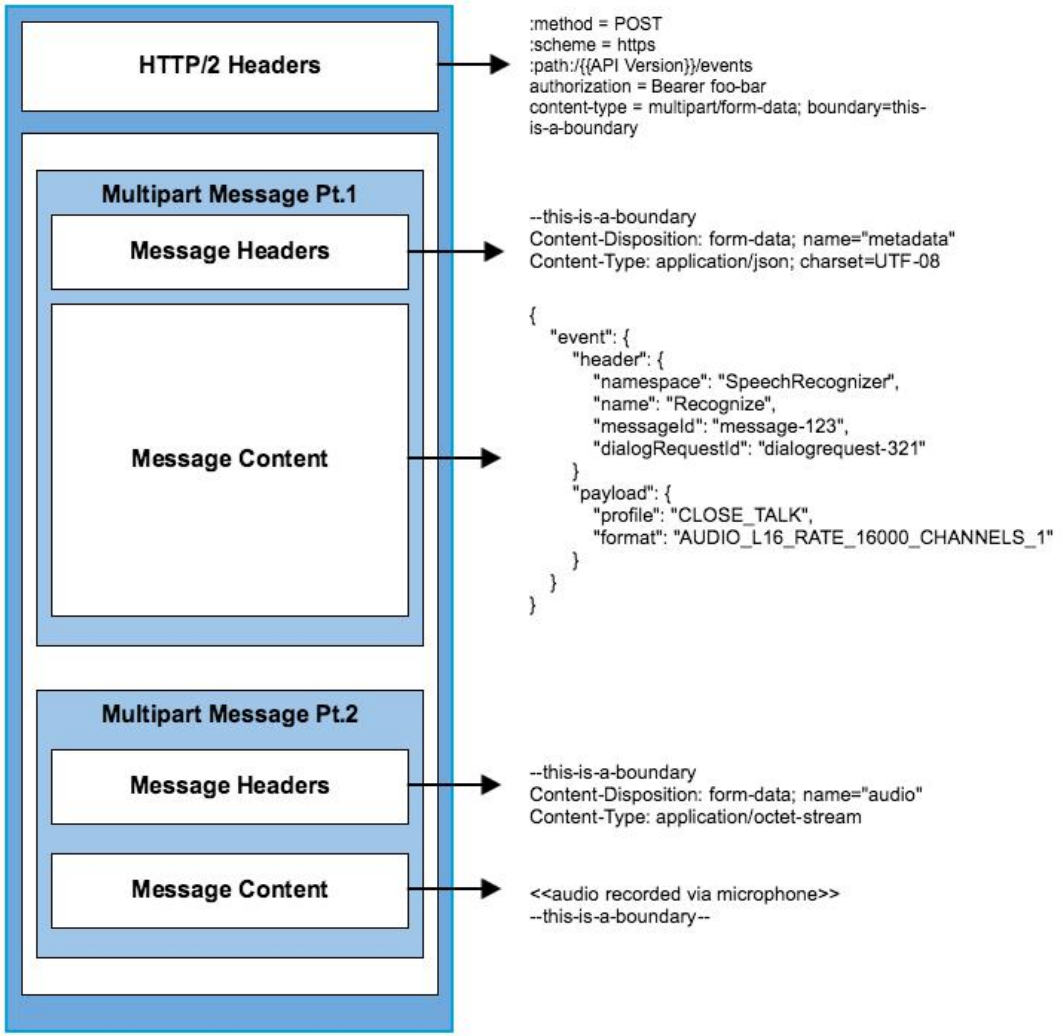
http 1.0中默认是关闭的，需要在http头加入"Connection: Keep-Alive"，才能启用Keep-Alive；http 1.1中默认启用Keep-Alive，如果加入"Connection: close "，才关闭。目前大部分浏览器都是用http1.1协议，也就是说默认都会发起Keep-Alive的连接请求了，所以是否能完成一个完整的Keep-Alive连接就看服务器设置情况。

半双工

长/短连接

2.1、Http请求结构


HTTP Request
or Response
HTTP Header
Body
Trailer
HTTP Header
Request or
Response Line
MIME Header
MIME Field
name:value
. . .
MIME Field
name:value


HTTP/2 Headers
:method = POST
:scheme = https
:path:/{{API Version}}/events
authorization = Bearer foo-bar
content-type = multipart/form-data; boundary=this-is-a-boundary
Multipart Message Pt.1
Message Headers
--this-is-a-boundary
Content-Disposition: form-data; name="metadata"
Content-Type: application/json; charset=UTF-08
Message Content
{
"event": {
"header": {
"namespace": "SpeechRecognizer",
"name": "Recognize",
"messageId": "message-123",
"dialogRequestId": "dialogrequest-321"
}
"payload": {
"profile": "CLOSE_TALK",
"format": "AUDIO_L16_RATE_16000_CHANNELS_1"
}
}
}
Multipart Message Pt.2
Message Headers
--this-is-a-boundary
Content-Disposition: form-data; name="audio"
Content-Type: application/octet-stream
Message Content
<<audio recorded via microphone>>
--this-is-a-boundary--

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

3.1、HttpURLConnection & Volley网络请求

```
class MyTask extends AsyncTask<String, Void, String> {

        @Override
        protected String doInBackground(String... params) {
            InputStream mInputStream = null;
            HttpURLConnection connection = getHttpUrlConnection(params[0]);
            String result = "";
            try {
                connection.connect();
                int statusCode = connection.getResponseCode();
                String response = connection.getResponseMessage();
                mInputStream = connection.getInputStream();
                InputStreamReader inputStreamReader = new InputStreamReader(mInputStream
                BufferedReader reader = new BufferedReader(inputStreamReader);
                StringBuffer sb = new StringBuffer();
                String line;
                while ((line = reader.readLine()) != null) {
                    sb.append(line + "\n");
                }

                result = "StatusCode: " + statusCode + "\n"
                        + "Response" + response + "\n"
                        + sb.toString();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return result;
        }

        @Override
        protected void onPostExecute(String s) {
            super.onPostExecute(s);
            tv.setText(s);
        }
    }

    private HttpURLConnection getHttpUrlConnection(String url) {
        HttpURLConnection connection = null;
        try {
            URL mUrl = new URL(url);
            connection = (HttpURLConnection) mUrl.openConnection();
            connection.setConnectTimeout(20000);
            connection.setReadTimeout(40000);
            connection.setRequestMethod("GET");
            connection.setRequestProperty("Content-Type", "application/json");
            connection.setRequestProperty("Accept", "application/json");
            connection.setRequestProperty("Charset", "utf-8");
            connection.setRequestProperty("Content-Length", "0");

        } catch (MalformedURLException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        return connection;
    }
```

```
new MyTask().execute(BASE_URL);
```

1、HttpURLConnection

```
protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mContext = this;
        queue = Volley.newRequestQueue(mContext);
        setContentView(R.layout.activity_http_volley_demo);
        tv = (TextView) findViewById(R.id.editText);

        final StringRequest request = new StringRequest(Request.Method.GET, BASE_URL,
                new ResponseSuccessListener(), new ResponseFailListener());
        findViewById(R.id.volley).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                queue.add(request);
            }
        });

    }

    private class ResponseSuccessListener implements com.android.volley.Response.Listene

        @Override
        public void onResponse(String response) {
            tv.setText(response);
        }
    }

    private class ResponseFailListener implements Response.ErrorListener {

        @Override
        public void onErrorResponse(VolleyError error) {
            Toast.makeText(mContext, error.toString(), Toast.LENGTH_SHORT).show();
        }
    }
```

2、Volley

3.2、Retrofit的使用

使用 Retrofit 的步骤共有7个：

步骤1： 添加Retrofit库的依赖

步骤2： 创建 接收服务器返回数据 的类

步骤3： 创建 用于描述网络请求 的接口

步骤4： 创建 Retrofit 实例

步骤5： 创建 网络请求接口实例 并 配置网络请求参数

步骤6： 发送网络请求（异步 / 同步）封装了 数据转换、线程切换的操作

步骤7： 处理服务器返回的数据

类型	注解名称	解释	作用域
网络请求方法	@GET	所有方法分别对应Http中的网络请求方法； 都接收一个网络地址URL（也可以不指定，通过@Http设置）	网络请求接口的方法
	@POST		
	@PUT		
	@DELETE		
	@PATH		
	@HEAD		
	@OPTIONS		
	@HTTP	用于替换以上7个注解的作用 及 更多功能拓展	
标记类	@FormUrlEncoded	表示请求体是一个Form表单	网络请求接口的方法
	@Multipart	表示请求体是一个支持文件上传的Form表单	
	@Streaming	表示返回的数据以流的形式返回； 适用于返回数据较大的场景； （如果没有使用该注解，默认把数据全部载入内存；之后获取数据也是从内存中读取）	
网络请求参数	@Headers	添加请求头	网络请求接口的方法
	@Header	添加不固定值的Header	网络请求接口的方法的参数 （如Call<> getCall(*)中的*）
	@Body	用于非表单请求体	
	@Field	向Post表单传入键值对	
	@FieldMap		
	@Part	用于表单字段； 适用于有文件上传的情况	
	@PartMap		
	@Query	用于表单字段； 功能同@Field与@FieldMap； （区别在于@Query和@QueryMap的数据体现在URL上，@Field与@FieldMap的数据体现在请求体上；但生成的数据是一致的。）	
	@QueryMap		
	@Path	URL 缺省值	
	@URL	URL设置	

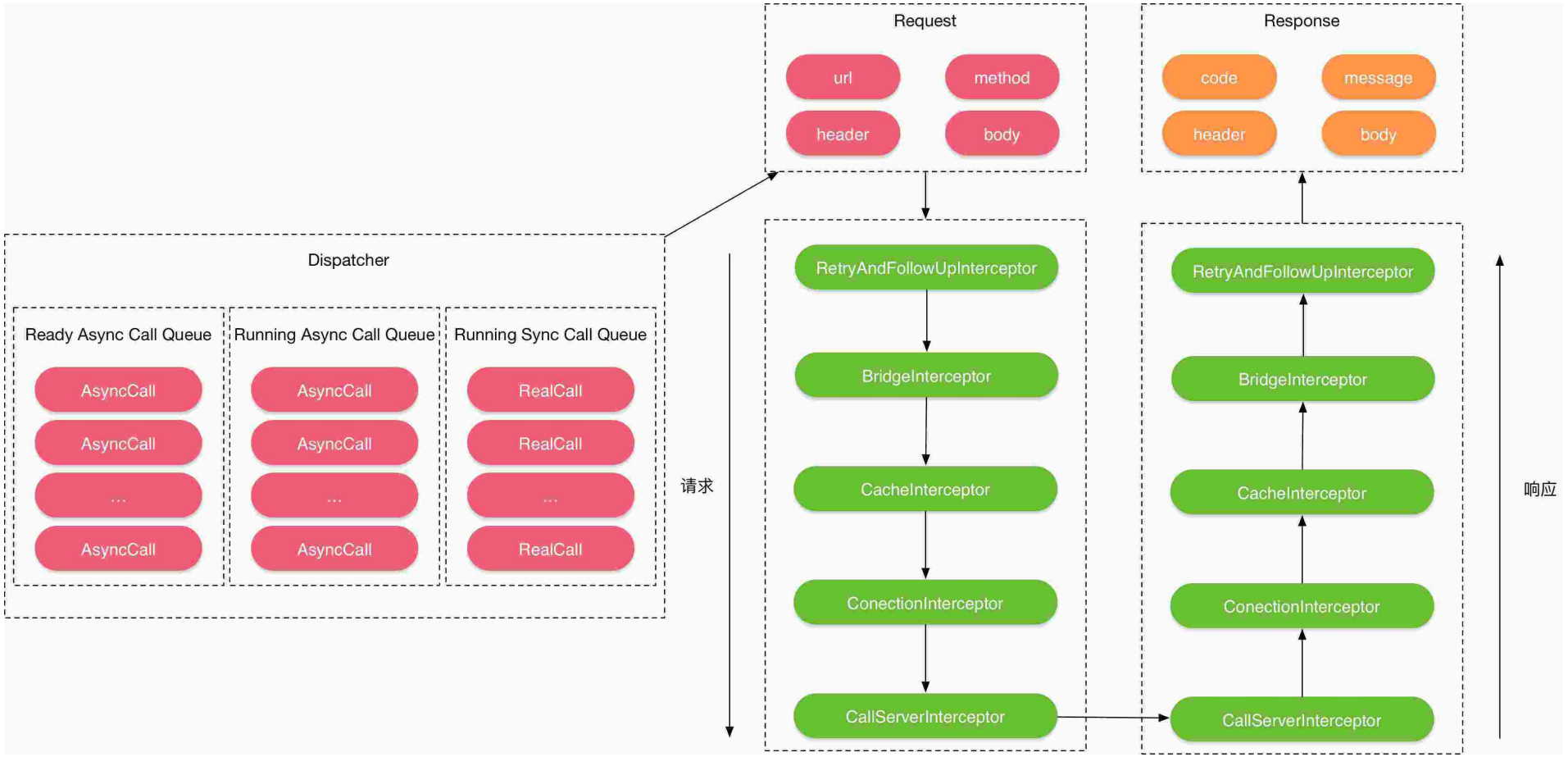
3.3、Retrofit的作用

1、请求参数通过注解代替

2、规范了请求的格式

3、相对于HttpURLConnection解决了出去的的问题

3.4 OkHttp原理


Request
url
method
header
body
Response
code
message
header
body
Dispatcher
Ready Async Call Queue
AsyncCall
AsyncCall
...
AsyncCall
Running Async Call Queue
AsyncCall
AsyncCall
...
AsyncCall
Running Sync Call Queue
RealCall
RealCall
...
RealCall
请求
RetryAndFollowUpInterceptor
BridgeInterceptor
CacheInterceptor
ConectionInterceptor
CallServerInterceptor
RetryAndFollowUpInterceptor
BridgeInterceptor
CacheInterceptor
ConectionInterceptor
CallServerInterceptor
响应

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

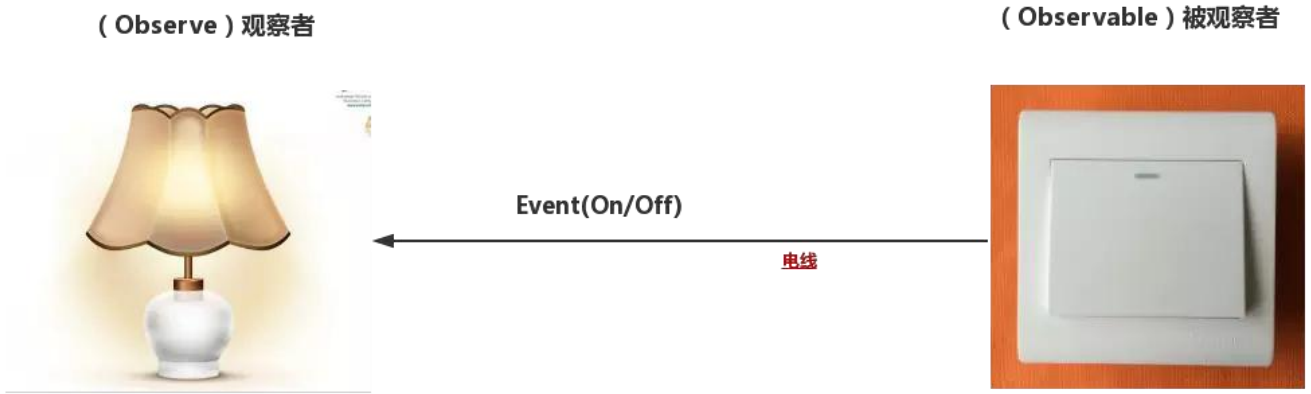
5、网络环境切换及网络层架构

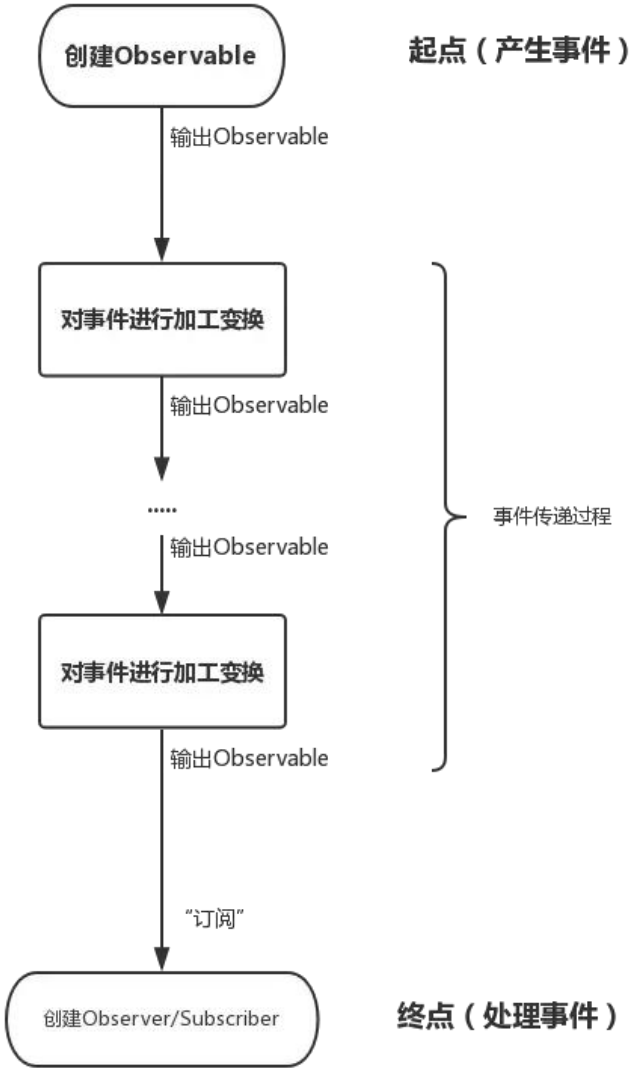
6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

4.1 Rxjava的原理





3.2、Rxjava的作用

1、方便数据回来的处理

2、过滤返回的数据，对错误进行统一处理

3、方便线程的切换

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

5.1 环境切换

本地开发　　集成测试　　正式发布

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

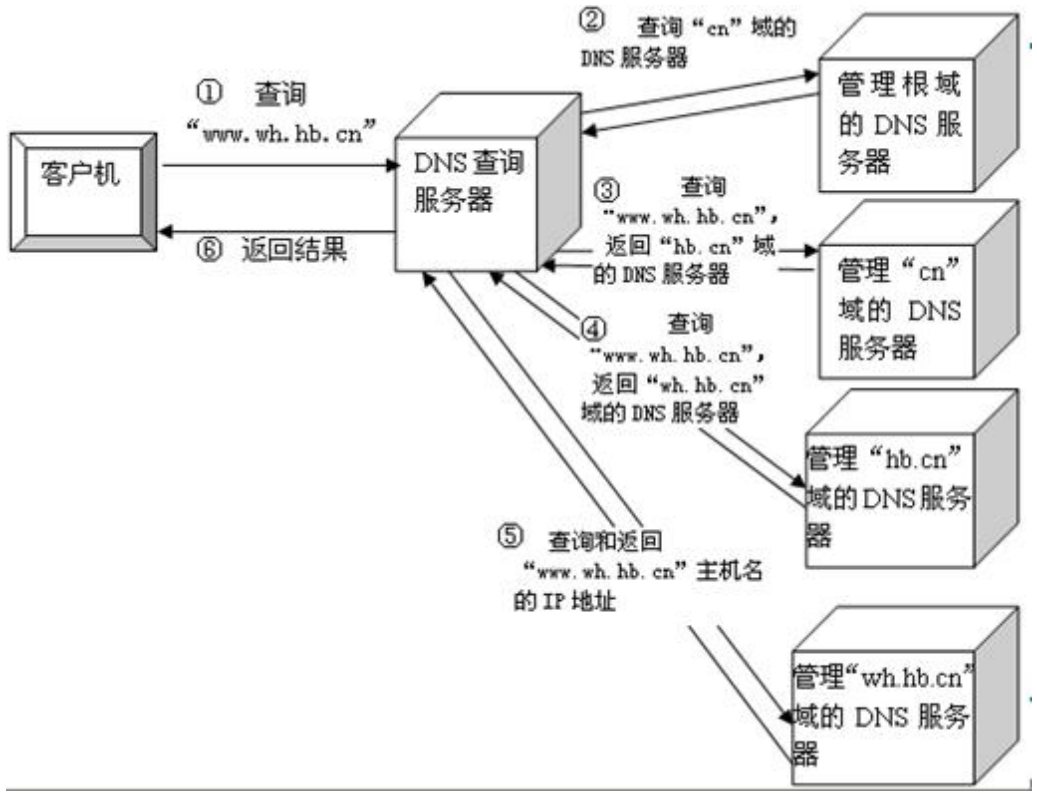
5、网络环境切换及网络层架构

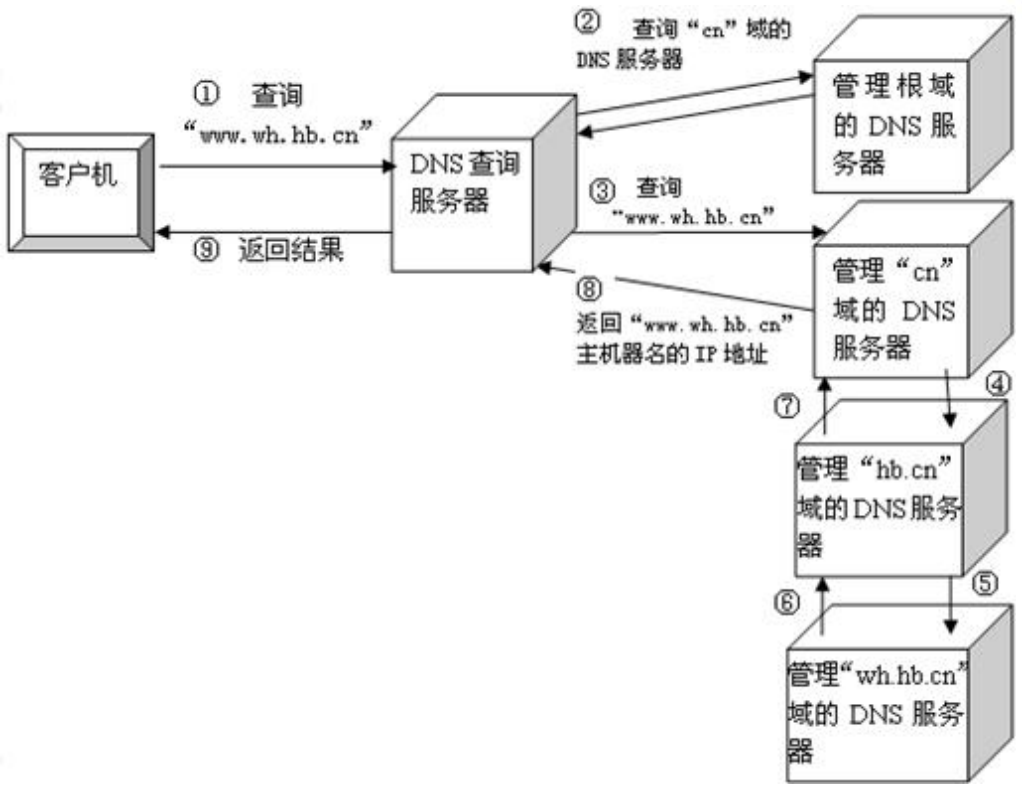
6、网络的可靠性保障

7、网络的安全性保障

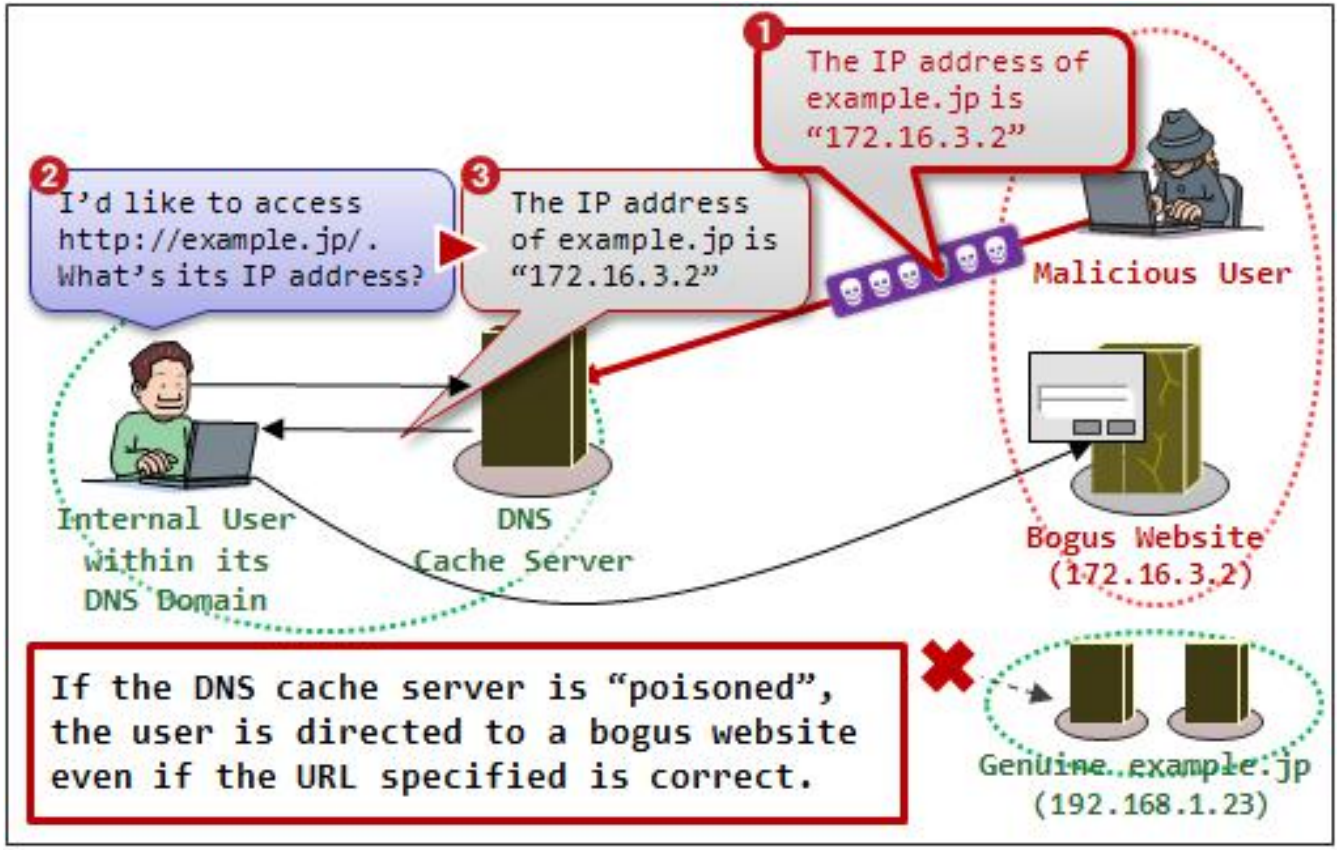
8、Android移动架构师知识体系构建

3.1 DNS流程


客户机
① 查询
“www.wh.hb.cn”
DNS查询服务器
② 查询“cn”域的DNS服务器
管理根域的DNS服务器
③ 查询“www.wh.hb.cn”，返回“hb.cn”域的DNS服务器
管理“cn”域的DNS服务器
④ 查询“www.wh.hb.cn”，返回“wh.hb.cn”域的DNS服务器
管理“hb.cn”域的DNS服务器
⑤ 查询和返回“www.wh.hb.cn”主机名的IP地址
管理“wh.hb.cn”域的DNS服务器
⑥ 返回结果


客户机
① 查询
“www.wh.hb.cn”
DNS查询服务器
② 查询“cn”域的DNS服务器
管理根域的DNS服务器
③ 查询
“www.wh.hb.cn”
管理“cn”域的DNS服务器
④
⑤
管理“hb.cn”域的DNS服务器
⑥
管理“wh.hb.cn”域的DNS服务器
⑦
⑧ 返回“www.wh.hb.cn”主机器名的IP地址
⑨ 返回结果

3.2 DNS缓存污染



网域服务器缓存污染（DNS cache pollution），又称**域名服务器缓存投毒**（DNS cache poisoning）、**DNS缓存投毒**，是指一些刻意制造或无意中制造出来的域名服务器数据包，把域名指往不正确的IP地址。一般来说，在互联网上都有可信赖的网域服务器，但为减低网络上的流量压力，一般的域名服务器都会把从上游的域名服务器获得的解析记录暂存起来，待下次有其他机器要求解析域名时，可以立即提供服务。一旦有关网域的局域域名服务器的缓存受到污染，就会把网域内的计算机导引往错误的服务器或服务器的网址。

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

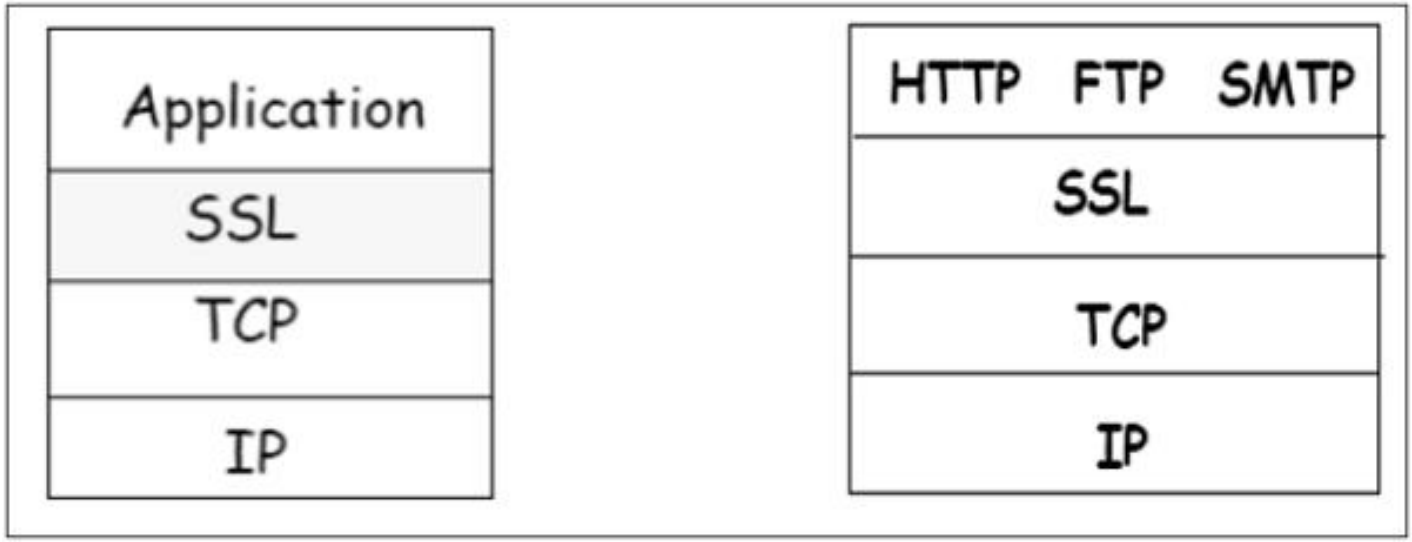
5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

8、Android移动架构师知识体系构建

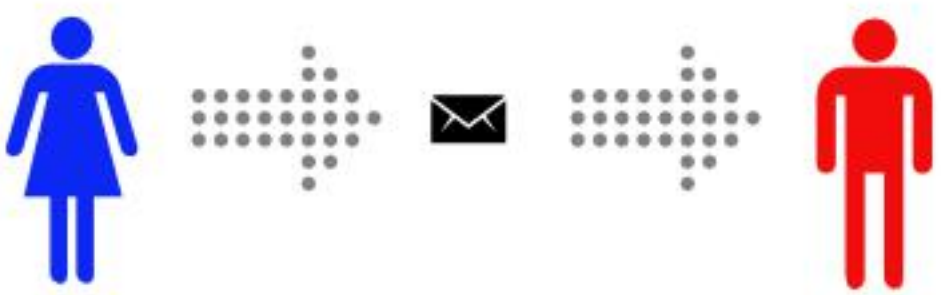
7.1 SSL/TLS介绍



（1） **窃听风险**（eavesdropping）：第三方可以获知通信内容。
（2） **篡改风险**（tampering）：第三方可以修改通信内容。
（3） **冒充风险**（pretending）：第三方可以冒充他人身份参与通信。

TLS以SSL 3.0为基础于1999年作为SSL的新版本推出。TLS协议和SSL 3.0之间的差异并不明显，但是他们都非常重要且TLS 1.0 和 SSL 3.0不具有互操作性

7.2 明文通信



1、无加密通信

2、通信被监听

7.3 对称加密

1、确定密钥

2、Alice使用密钥加密消息

3、Alice通过网络发送加密消息给Bill

4、Bill使用密钥解密加密消息

7.4 非对称加密

1、Bill买了锁和钥匙

2、 Bill把锁发给Alice，钥匙自己拿着

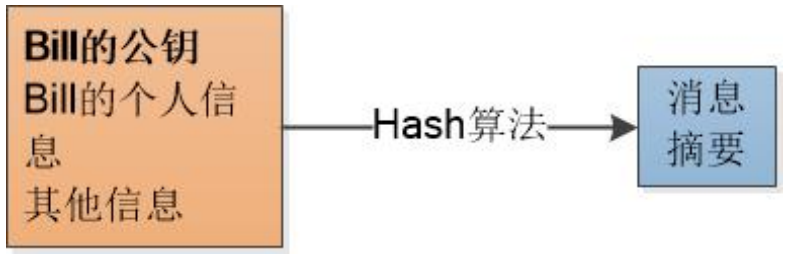
3、Alice用Bill的锁把信息锁起来

4、Alice把用Bill的锁锁起来的信息发给Bill

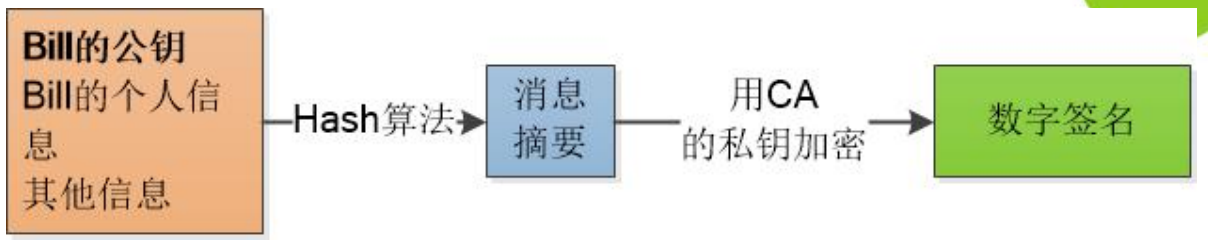
5、 Bill用钥匙打开锁拿到信息

公钥怎么保证安全？

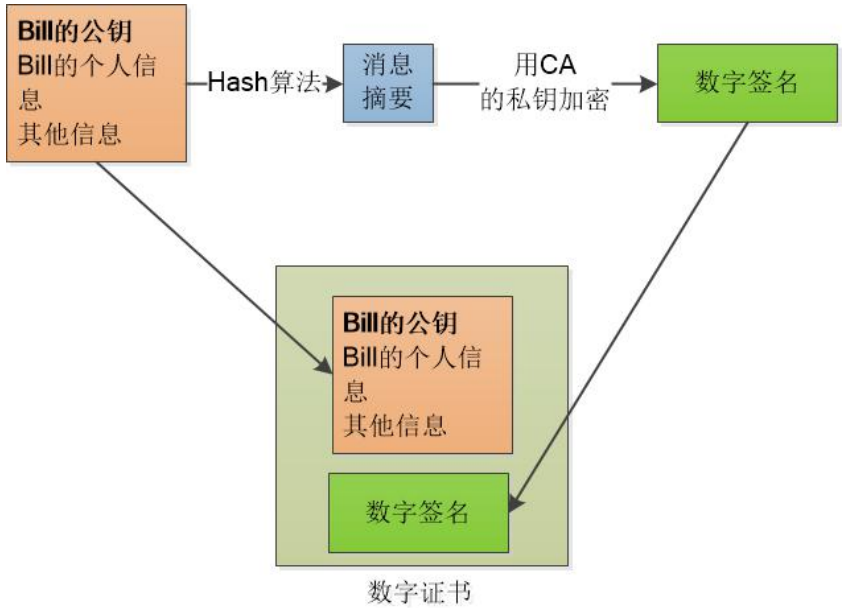
7.5 消息摘要/数字证书/CA



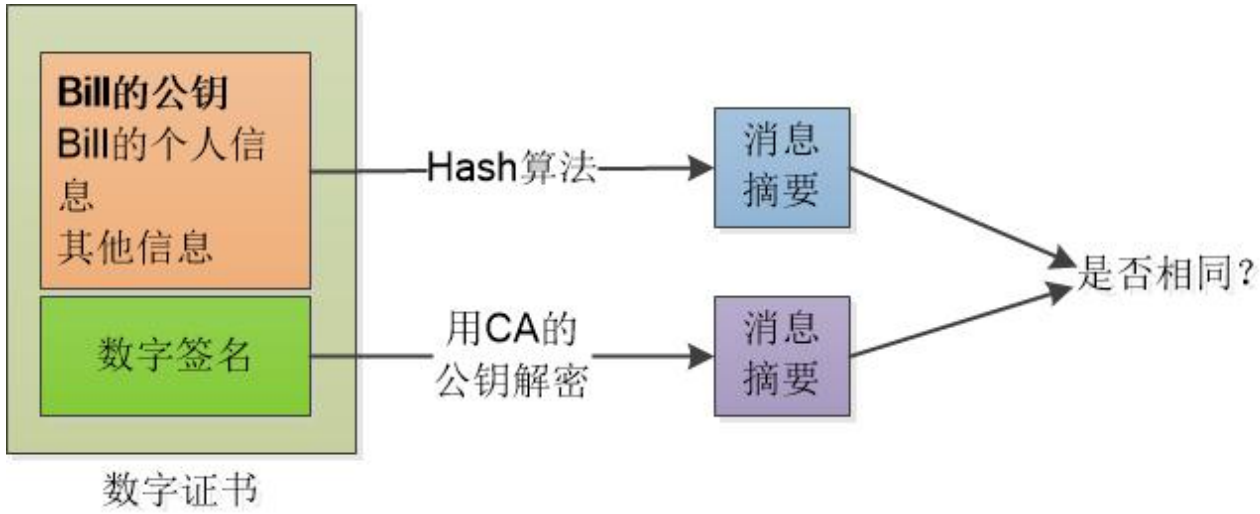
1、消息摘要



2、数字签名

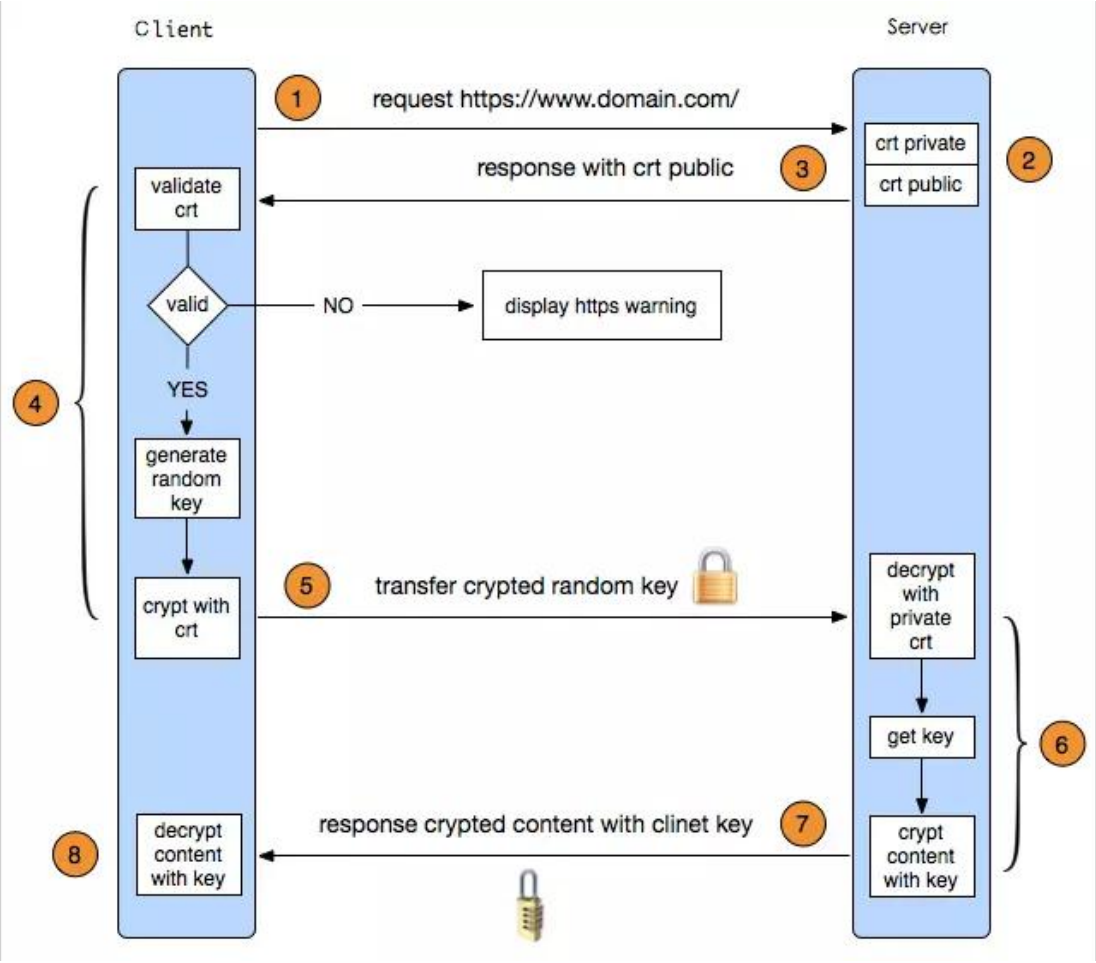


3、数字证书



4、验证

7.6 Https通信流程



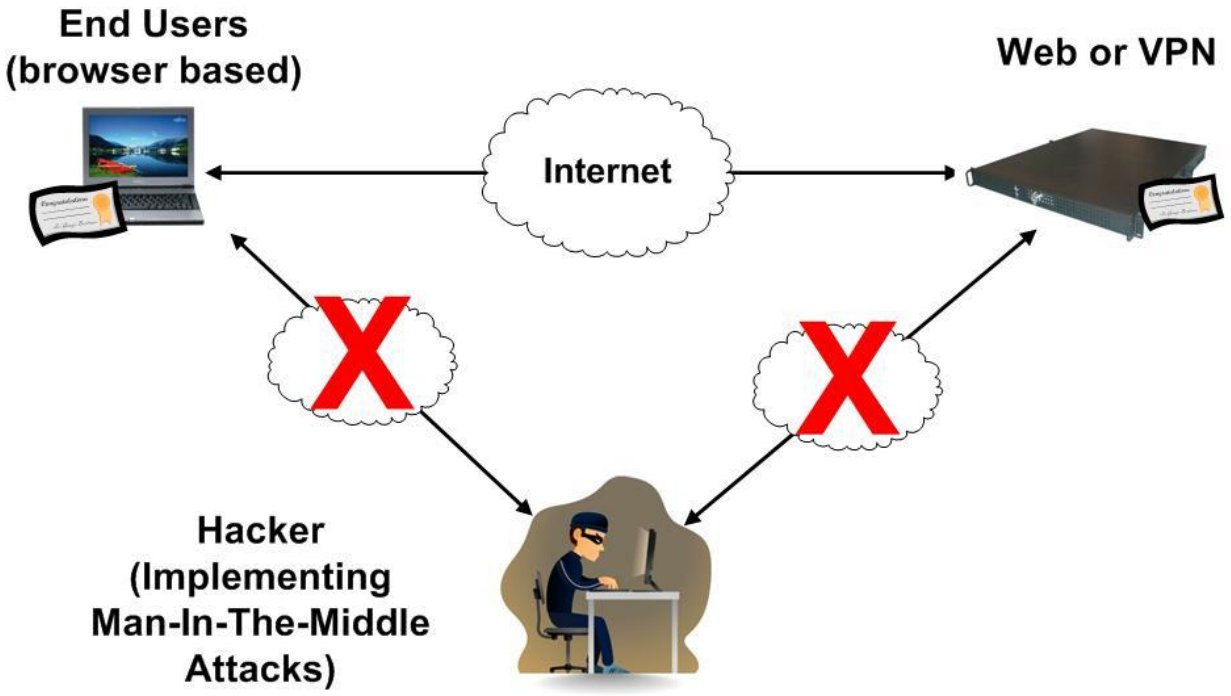
一个HTTPS请求实际上包含了两次HTTP传输，可以细分为8步：

1. 客户端向服务器发起HTTPS请求，连接到服务器的443端口。
2. 服务器端有一个密钥对，即公钥和私钥，服务器端保存着私钥，不能将其泄露，公钥可以发送给任何人。
3. 服务器发送了一个SSL证书给客户端，SSL 证书中包含的具体内容有：证书的发布机构CA、证书的有效期、公钥、证书所有者、签名。
4. 客户端收到服务器端的SSL证书之后，验证服务器发送的数字证书的合法性，如果发现发现数字证书有问题，那么HTTPS传输就无法继续。如果数字证书合格，那么客户端会生成一个随机值，这个随机值就是用于进行对称加密的密钥；然后用公钥对对称密钥进行加密，变成密文。至此，HTTPS中的第一次HTTP请求结束。
5. 客户端会发起HTTPS中的第二个HTTP请求，将加密之后的客户端密钥发送给服务器。
6. 服务器接收到客户端发来的密文之后，会用自己的私钥对其进行非对称解密，解密之后的明文就是对称密钥，然后用对称密钥对数据进行对称加密。
7. 服务器将加密后的密文发送给客户端。
8. 客户端收到服务器发送来的密文，用对称密钥对其进行对称解密，得到服务器发送的数据。这样HTTPS中的第二个HTTP请求结束，整个HTTPS传输完成。

中间人攻击怎么办？

7.7 中间人攻击

中间人攻击（英语：Man-in-the-middle attack，缩写：MITM）是指攻击者与通讯的两端分别创建独立的联系，并交换其所收到的数据，使通讯的两端认为他们正在通过一个私密的连接与对方直接对话，但事实上整个会话都被攻击者完全控制。



1. 服务器向客户端发送公钥。
2. 攻击者截获公钥，保留在自己手上。然后攻击者自己生成一个【伪造的】公钥，发给客户端。
3. 客户端收到伪造的公钥后，生成加密hash值发给服务器。
4. 攻击者获得加密hash值，用自己的私钥解密获得真秘钥。同时生成假的加密hash值，发给服务器。
5. 服务器用私钥解密获得假秘钥。

7.8.1 Android应用怎样防止MITM？

1、千万不要在X509TrustManager里不对客户端和服务器端的证书不做校验

```
new X509TrustManager() {
        public void checkClientTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
          //do nothing，接受任意客户端证书
        }

        public void checkServerTrusted(X509Certificate[] chain, String authType)
            throws CertificateException {
          //do nothing，接受任意服务端证书
        }

        public X509Certificate[] getAcceptedIssuers() {
          return new X509Certificate[0];
        }
};
```

android:networkSecurityConfig="@xml/network_security_config"

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
  <base-config>
    <trust-anchors>
      <certificates src="..."/>
      ...
    </trust-anchors>
  </base-config>

  <domain-config>
    <domain>android.com</domain>
    ...
    <trust-anchors>
      <certificates src="..."/>
      ...
    </trust-anchors>
    <pin-set>
      <pin digest="...">...</pin>
      ...
    </pin-set>
  </domain-config>
  ...
  <debug-overrides>
    <trust-anchors>
      <certificates src="..."/>
      ...
    </trust-anchors>
  </debug-overrides>
</network-security-config>
```

7.8.2 Android应用怎样防止MITM？

2、CA不在Android系统的列表怎么办？

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
   <domain-config>
      <domain includeSubdomains="true">example.com</domain>
      <trust-anchors>
         <certificates src="@raw/my_ca"/>
      </trust-anchors>
   </domain-config>
</network-security-config>
```

3、自签名证书怎么办？

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
   <base-config cleartextTrafficPermitted="true" />
   <domain-config>
      <domain includeSubdomains="true">httpbin.org</domain>
      <pin-set expiration="2020-03-07">
         <pin digest="SHA-256">Yvh6l+lXgqrBJrCtxwr9r/vbERE37/5/p6AaRRsiboQ=</pin>
      </pin-set>
   </domain-config>
</network-security-config>
```

1、网络模块在应用中的地位

2、Http协议

3、Retrofit实现网络请求

4、Rxjava原理和Retrofit的结合

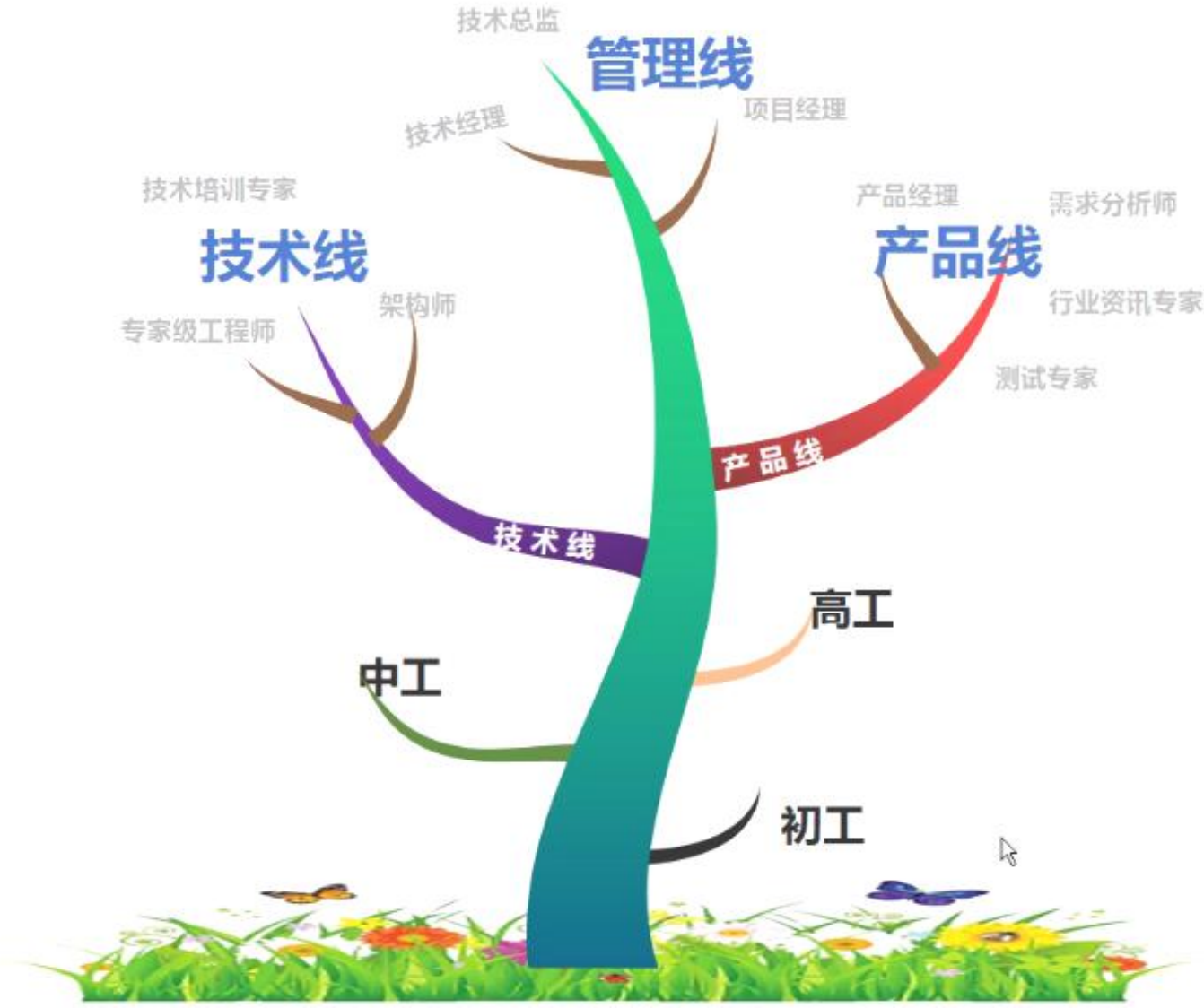
5、网络环境切换及网络层架构

6、网络的可靠性保障

7、网络的安全性保障

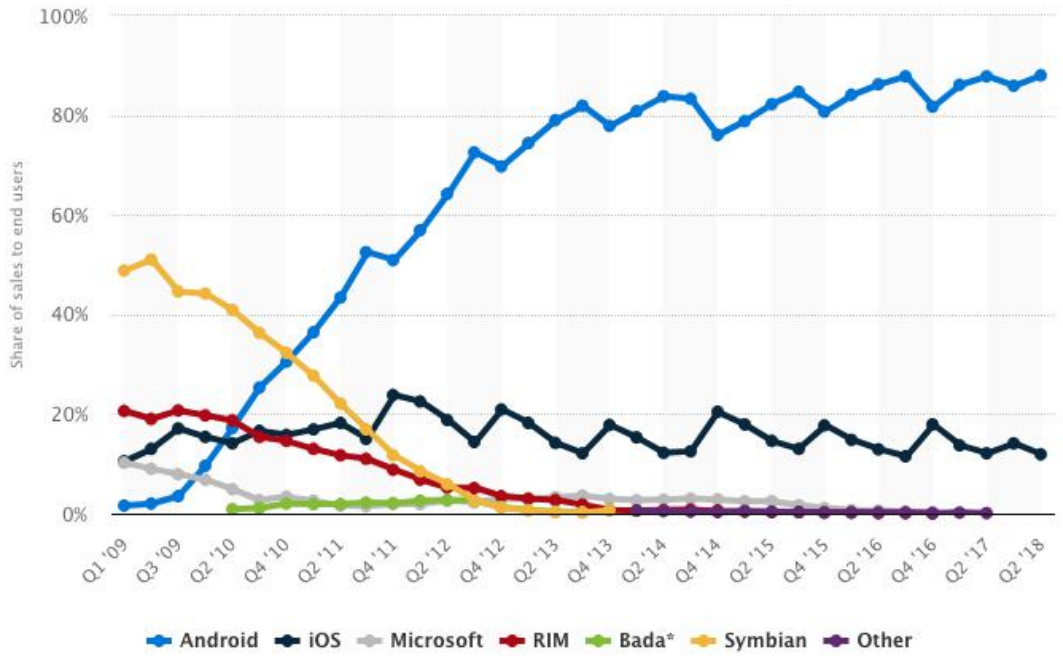
8、Android移动架构师知识体系构建

8.1 Android程序员职业发展规划



国家发展 → 行业发展 → 自身发展

8.2 Android移动架构师的未来



面向用户 面向终端 利于创业

移动端

谢谢大家