一种安卓应用深度链接的云端执行框架

张化龙[1] **马郓**[1,2] **柳熠**[1] **刘譞哲**[1+]
（北京大学高可信软件技术教育部重点实验室 北京 100871）[1]
（清华大学软件学院 北京 100084）[2]

摘 要 移动深度链接是一种通过 URI（Uniform Resource Identifier）从移动应用外部直接访问内部特定页面的机制。目前深度链接的执行方式有直接执行和延迟执行两种。然而，随着移动应用市场的发展，深度链接启动的目标应用经常未在手机上安装，限制了深度链接机制的应用和推广，给用户的使用造成不便。因此设计并实现了一种深度链接的云端执行框架：在云端安卓虚拟机中执行深度链接，手机通过服务器与云端虚拟机进行通信，进而在手机未安装目标应用时也能达到与本地直接执行一致的用户体验。实验表明，该框架可以正确地执行深度链接，并且延迟及流量消耗处于可控范围内。

关键词 安卓应用，深度链接，云端执行

中图法分类号 TP311 **文献标识码** A **DOI** （投稿时不提供 DOI 号）

A Framework of Cloud-based Execution of Deep Linking for Android Applications

ZHANG Hua-long[1] MA Yun[1,2] LIU Yi[1] LIU Xuan-zhe[1+]
(Key Lab of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing 100871, China)[1]
(School of Software, Tsinghua University, Beijing 100084, China)[2]
(Corresponding author: E-mail: liuxuanzhe@pku.edu.cn)[+]

Abstract Mobile deep linking is a mechanism which enables targeting and opening a specific page of an app externally with an accessible URI. So far, there are two mechanisms to execute deep linking of Android applications: direct execution and indirect execution. However, with the development of the mobile application market, the cases with deep linking's target application not installed increase and the expense of the installation of unknown applications rise. This paper designs and implements a framework of cloud-based execution of deep linking. With the communication between the phone and the virtual machine on the server, users don't need to install the target application, and can enjoy the same user experience with cloud-based execution as direct execution. The experiment shows that we can execute deep linking correctly using cloud-based execution with acceptable execution time and traffic consumption.

Keywords Android application, Deep linking, Cloud-based execution

1 引言

目前，移动设备已经超过 PC 成为第一大上网终端。在移动设备上，用户更加偏好使用 APP 来满足其消息浏览和功能使用的需要。Flurry 2015 年的数据显示，移动用户 86%的使用时间都是在各个 APP 中[1]。为了完成特定的任务，用户经常需要在不同 APP 之间进行切换。例如用户在朋友圈中，看到一篇关于某部电影的文章，他可能需要打开豆瓣应用浏览该电影的影评，进而打开购票软件网上购票，最后使用打车软件前

到稿日期： 返修日期：
本文受国家高技术研究发展计划(863 计划)(2016AA01A202)，国家重点基础研究发展计划(973 计划)(2016CB320703)，国家自然科学基金(F020208)基金资助。张化龙（1995-），男，硕士研究生，主要研究方向为 Web，软件工程等；马郓（1989-），男，博士，助理研究员，主要研究方向为移动计算，Web 和服务计算；柳熠（1993-），男，博士研究生，主要研究方向为移动计算，服务计算和移动 Web；刘譞哲（1980-），男，博士，副教授，主要研究方向为移动计算，Web 技术和大数据等，E-mail: liuxuanzhe@pku.edu.cn(通信作者)。

往影城。为了支持用户在不同应用的页面之间进行跳转，移动深度链接机制近年来被各大移动操作系统和互联网服务提供商所提出和支持。类似 Web 网页之间的超链接，深度链接是应用中一种通过 URI 从外部直接访问内部特定页面的机制[2]。

在移动操作系统上，深度链接是利用系统提供的应用间通信接口来实现的。以安卓系统为例，深度链接用隐式 Intent 实现，创建一个 Intent，并且指定目标应用的包名和参数等，就可以打开目标应用的目标页面。目前，深度链接的执行方式主要有直接执行和延迟执行两种：直接执行要求手机中已安装目标应用，系统会根据深度链接直接打开目标页面；延迟执行会在手机上先安装目标应用然后再打开目标页面。然而，随着移动应用市场的进一步发展，深度链接面临新的挑战：

1）目前手机用户应用的安装数量表现出长尾分布[3]，大部分手机用户安装的应用相当少。因此随着应用中深度链接的广度增加，深度链接启动的目标应用并未安装的情况会越来越多，直接执行适用的情况变少，必须使用延迟执行，造成用户需要等待应用安装之后才能使用，影响用户体验。

2）随着市场上应用数量不断增加，其质量越发良莠不齐，用户越来越倾向于拒绝安装未知应用。尤其是在执行深度链接时，用户往往不希望仅为了一次使用其中的某个功能而下载安装一个应用。

针对上述挑战，本文设计并实现了一种安卓应用深度链接的云端执行框架：手机无需安装目标应用而是在云端安卓虚拟机中执行深度链接，并且手机与云端虚拟机进行通信，获得目标页面供用户使用，同时在用户使用时维护手机和云端虚拟机的状态同步，进而在手机未安装应用时也能达到与直接执行一致的用户体验，免去延迟执行的安装步骤。

本文后续的组织结构如下：第二部分对深度链接云端执行框架进行了分析与设计；第三部分介绍了面向安卓应用的深度链接云端执行框架的具体实现；第四部分对云端执行框架进行了实验验证，检测了执行结果的正确性，测量了云端执行的效率以及开销等；第五部分介绍相关工作；最后对全文的工作进行总结，并提出未来工作的几个方向。

2 云端执行框架的分析与设计

2.1 云端执行框架的分析

针对目前深度链接的执行面临的挑战，用户手机上应用的安装数较少，并且倾向于拒绝安装未知应用，当用户执行深度链接而目标应用未安装时，我们可以在云端的安卓虚拟机中执行该深度链接，执行完成后，用户手机通过与云端虚拟机的通信来获得目标页面以供使用。这样就免去了安装应用的步骤，而达到了和直接执行一致的用户体验。

在云端执行过程中，由于服务器需要处理多个用户发送的请求，因此需要给用户手机一个身份标识，用户使用该标识申请云端执行服务。为达到与直接执行一致的用户体验，用户手机在执行完深度链接后，应该得到一个可交互的界面，在得到的交互界面上可以进行与真实页面一样的操作。当用户操作完毕后，需要告知服务器深度链接执行完毕。由此可以得到云端执行框架需要满足如下几个需求：

1）身份认证：用户手机可以申请一个标识身份的 Token 字符串，并在发送其他请求时带上身份标识。服务器可以合理分配 Token，并正确验证请求中的 Token。

2）实时状态同步：为了得到与直接执行一致的用户体验，在用户操作交互界面时，需要保持用户手机与云端虚拟机的实时状态同步，包括用户操作的页面，登陆的用户信息等，进而得到与真实页面一致的体验。

3）并发处理：用户手机请求执行深度链接时，服务器及时响应，并为其分配空闲的虚拟机；用户手机关闭交互界面时，

服务器及时停止对应的深度链接，释放虚拟机资源。

根据上述分析，可以得到云端执行框架的主要功能模块：用户手机上需要部署一个客户端，用于与服务器进行通信；服务器需要监听用户的请求，并根据请求进行调度，选择一台安卓虚拟机提供服务，然后控制该虚拟机执行深度链接，执行完成后使其与用户手机通信保持状态同步；云端安卓虚拟机需要部署一个客户端，暴露一个接口以供服务器对其进行控制。

2.2 云端执行框架的设计

根据云端执行框架的需求及主要功能模块，结合具体实现环境，得到如下整体架构图。

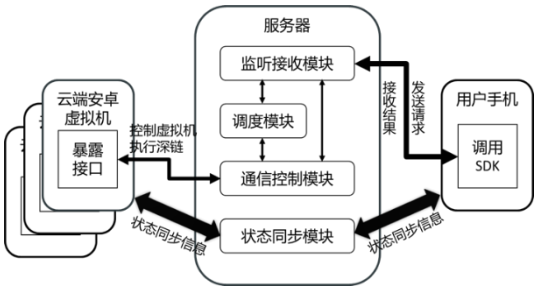

图 1 整体架构图

如图 1 所示，云端执行框架的整体架构主要包括用户手机，服务器和云端安卓虚拟机三个部分。

云端安卓虚拟机是部署好的一些用以提供服务的虚拟机。每个虚拟机上会安装数个 APP 的某个版本，并在初始化时将本机的 ID 及支持的深度链接种类告知服务器的调度模块。虚拟机并不维护应用数据，若应用版本更新，则手动更新虚拟机上的应用。同时部署一个客户端，暴露一个接口用于服务器对其进行操作及返回操作结果。在通过该接口接收到服务器的控制信息时，根据参数执行相应的深度链接。深度链接种类以 APP 名+目标页面名表示，执行时以隐式 Intent 的形式跳转到指定的页面。当需要执行的深度链接种类即需要安装的应用数量过多时，增加新的虚拟机以保证性能。

用户手机上需要部署一个客户端，通过该客户端可以与服务器进行通信。同时需要提供两种调用接口，分别为获取 Token 和执行深度链接。在用户手机调用接口时，会启动客户端，根据调用接口的参数，向服务器发送请求。若调用的接口为获取 Token，则等待从服务器接收结果即可；若调用的接口为执行深度链接，则在收到结果信息后，打开一个新的 activity 即交互界面，通过服务器的状态同步模块与云端虚拟机保持状态同步。在该界面里会提供一个按钮，点击后会关闭交互界面，并告知服务器本次深度链接执行完毕。

服务器主要包括四个模块：

1）监听接收模块的主要功能为等待接收并处理用户手机发送的请求。若请求类型为执行深度链接，则通过调度模块决定提供服务的虚拟机 ID，然后通过通信控制模块进行控制；若请求类型为结束深度链接，则告知调度模块对应的深度链接完成，同样通过通信控制模块进行控制。

2）调度模块的主要功能为根据深度链接的种类，决定选择哪台虚拟机来提供服务。调度模块中维护了一个列表，列表记录所有虚拟机的状态及其支持的深度链接种类。在有请求时选择一台支持目标深度链接种类的空闲虚拟机置为繁忙，在深度链接执行完毕后将其恢复为空闲。

3）通信控制模块的主要功能为控制安卓虚拟机，通过其暴露的接口向其发送控制信息并接收执行结果，包括执行深度链接和关闭当前深度链接。另外还负责接收虚拟机初始化时发送过来的信息，并在处理后告知调度模块。

4）状态同步模块的主要功能为提供用户手机与虚拟机保持实时状态同步的服务。该服务部署在服务器上，会提供与虚拟机通信的接口。在执行深度链接成功后，用户手机访问该服务，跳转到执行深度链接的虚拟机对应的接口实现状态同步。在深度链接执行完毕后，停止对用户手

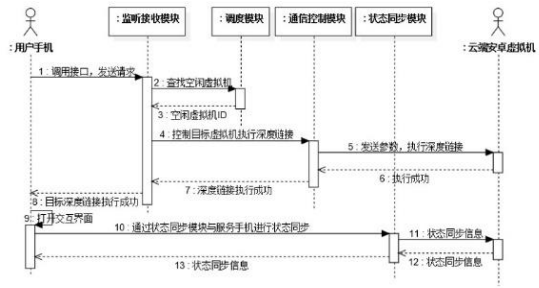

图 2　深度链接的执行流程

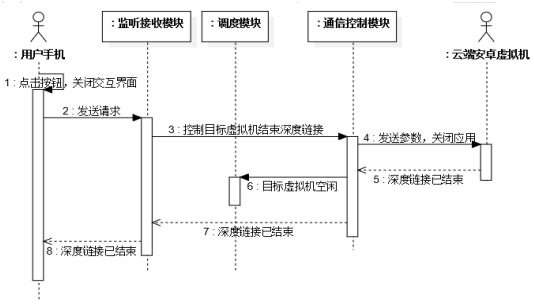

图 3　交互界面的关闭流程

机的服务。

图 2 展示了用户手机请求执行深度链接的流程。首先是用户手机调用接口，向服务器发送请求。服务器的监听接收模块收到该请求后，根据参数中的深度链接的类型查询调度模块来获取一个空闲云端安卓虚拟机的 ID。调度模块根据深度链接类型查找虚拟机列表，尝试选择一个支持该类型的空闲虚拟机：若无空闲则本次执行失败，将失败信息返回用户手机；若有空闲则将该虚拟机置为繁忙，将其 ID 返回给监听模块然后告知通信控制模块。通信控制模块根据 ID 找到目标虚拟机，将深度链接的参数发送过去。虚拟机接收到后，根据深度链接类型中的 APP 名和目标页面名执行相应的深度链接，得到返回信息，发送给通信控制模块，然后进一步返回给监听模块。监听模块收到该信息后，将提供服务的虚拟机的 ID 附带上一起返回给用户手机。用户手机收到后，打开一个交互界面，连接上服务器的状态同步模块，根据 ID 找到目标虚拟机，然后通过该模块维护状态同步，进行进一步的交互操作。

图 3 展示了当深度链接执行完毕、用户关闭交互界面的流程。首先是用户手机点击交互界面上的按钮，关闭交互界面，并向服务器发送结束深度链接的请求。服务器的监听接收模块收到该请求后，查询得到需要结束深度链接的目标虚拟机的 ID，将 ID 和深度链接跳转的目标应用的包名告知通信控制模块。通信控制模块根据 ID 找到目标虚拟机，将包名发送过去。虚拟机接收到后，强制关闭目标应用，结束深度链接后，将信息返回给通信控制模块。通信控制模块告知调度模块，将目标虚拟机的状态变为空闲，然后将深度链接已结束的信息通过监听模块返回给用户手机。

3　云端执行框架的实现

由于目前安卓应用中深度链接的执行方式由调用方决定，因此本文将云端执行框架实现为一组供深度链接调用方的开发者使用的 SDK。接下来先介绍实现过程中涉及的相关工具，然后介绍各个主要部分的具体实现。

3.1　本文使用的相关工具

3.1.1　STF

为了做到实时的状态同步，采用视频流加热点的方式来实现交互界面，参考 STF 来实现服务器上搭建的为用户手机与虚拟机提供状态同步的服务。STF(openstf)[4]全称为 Smartphone Test Farm，这是一个基于 nodejs+jade 模块开发的 Android 手机的远程控制平台，它可以在 Web 端对大量 Android 设备进行批量控制和管理，数量上限为 160 个左右。其优点有很多，例如通过服务器与手机的实时通信来确保屏幕的实时同步等。本文主要利用其作为 WEB 端工具的特点，从而可以不用在用户手机上安装任何应用，直接在 WebView 中访问网页就可以通过服务器与云端虚拟机同步。图 4 为 STF 的效果展示。

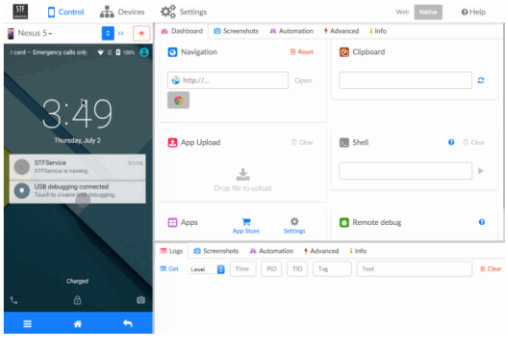

图 4　STF 效果图

3.1.2 Netty

为了简化系统实现，同时提高效率，本文采用了 Netty 框架来实现网络通信。这是一个基于 Java NIO[5]的 java 开源客户端/服务器端编程框架，是一款异步的、事件驱动的网络应用框架和工具，用于快速开发高性能、高可靠性的网络服务器和客户端[6]。其最大的优点是支持快速、简单地开发网络应用，简化和流线化了网络应用的编程开发过程，在保证易于开发的同时还保证了其应用的性能，稳定性和伸缩性。

本文在服务器和用户手机之间及服务器和云端安卓虚拟机之间，分别用 Netty 框架搭建了一对服务器端/客户端，用以进行网络通信。

3.2 用户手机

用户手机的实现主要包括两个部分，分别是 Netty 在客户端的部署代码以及交互界面的实现，下面分别介绍这两个部分。

3.2.1 Netty 客户端

Netty 在客户端的部署代码主要实现了三个函数，其中前两个为 SDK 接口的实现。三个函数大体相似，初始化客户端，向服务器发送请求，等待收到返回信息后关闭客户端。为了在关闭交互界面时，服务器可以定位到需要结束的深度链接，需要将 Token 和 APP 包名传给服务器。因此在执行深度链接对应的接口函数中，除了要在执行成功后打开交互界面外，还需要将 Token 和深度链接指向的 APP 包名记录下来。

3.2.2 交互界面

该部分实现了深度链接执行完成后打开的新的 activity。由于本文使用了 STF 作为保持状态同步的工具，而 STF 是一个 Web 端工具，因此直接在 activity 中打开一个 WebView，访问服务器上的 STF 服务，根据虚拟机 ID 找到提供服务的虚拟机页面。在 activity 中设置了一个悬浮按钮，点击后调用关闭交互界面函数。

3.3 安卓虚拟机

安卓虚拟机的实现主要包括 Netty 在客户端的部署代码，与用户手机上的客户端不同，该客户端在虚拟机配置完成后长期启动。为了保持调度模块中虚拟机配置信息的正确性，在客户端启动并连接到服务器后，立刻将虚拟机 ID 以及本机支持的深度链接种类发送给服务器，在关闭客户端之前，告知服务器该虚拟机停止服务。当收到结束当前深度链接的控制信息时，直接强制关闭目标应用，为下一次深度链接执行做准备。目前虚拟机在初始化时会预置好一个 APP，并以其中的深度链接作为自己支持的深度链接种类，并未对虚拟机上安装的应用做动态调度。

3.4 服务器

首先给出服务器接收到用户发送的执行深度链接的请求后，如何根据请求选择相应的虚拟机执行深度链接并获取结果的伪码。然后分别对服务器四个模块的实现进行介绍。

```
1. //监听接收模块
2.    receive msg//收到请求并处理
3.    test Token//检测 Token
4.    get freeVDID//根据深链种类获取空闲虚拟机 ID
5.    //调度模块
6.       get clientlist//获取支持该深链的虚拟机
7.       get freeclient//获取空闲虚拟机
8.       set busy//置为繁忙并返回其 ID
9.    save Token&VDID//保存 Token 及 ID
10.   send controlmsg//向目标虚拟机发送控制信息
11.   //通信控制模块
12.      get Channel//根据 ID 得到通信接口
13.      send msg//发送执行深度链接信息
14.      receive result//接收并返回执行结果
```

图 5 服务器处理执行深度链接的请求代码

3.4.1 监听接收模块

该模块主要包括两个部分，分别是与用户手机上的 Netty 客户端对应的服务器端的部署代码，以及负责保存 Token 并处理 Token 请求的部分。

其中 Netty 在服务器端的部署代码主要负责对服务器端进行初始化，然后不断创建子线程对收到的请求进行处理。本文目前处理请求的方式为简单的先来先服务，即将收到的请求保存入一个阻塞队列，服务器从该队列中依次取出消息进行处理。为了可以根据 Token 定位到需要关闭的虚拟机，在请求为执行深度链接时，将 Token 和提供服务的目标虚拟机 ID 保存下来，在请求为

结束深度链接时，根据 Token 查找到提供服务的虚拟机 ID。

3.4.2 通信控制模块

该模块主要是与安卓虚拟机上的 Netty 客户端对应的服务器端的部署代码部分，主要提供控制虚拟机的函数供监听模块调用，并处理收到的信息。其收到的信息可能是深度链接的执行结果，新虚拟机的登录或虚拟机停止服务。为了可以根据虚拟机 ID 定位到其对应的通信接口，该模块中维护了一个虚拟机 ID 和 Channel 的映射。若收到新虚拟机的登录或停止服务信息，则更新映射，然后告知调度模块对应的登录或停止服务信息。

3.4.3 调度模块

该模块主要负责维护虚拟机状态并为请求分配提供服务的虚拟机。将每一台安卓虚拟机抽象化为一个对象，记录包括虚拟机 ID，目前状态，以及支持的深度链接种类等信息。该模块维护了两个映射 appDevices 和 clients，前者对于每一种虚拟机支持的深度链接，记录了支持该种深度链接的所有虚拟机对应的 Client 对象，后者记录了虚拟机 ID 和 Client 对象的映射关系。当增加新的虚拟机时，根据 ID 和支持的深度链接种类，创建一个新的 Client 对象，状态为空闲，更新两个映射；当关闭虚拟机时，根据 ID 更新两个映射；当查找空闲虚拟机时，根据深度链接的种类，查找 appDevices 获得所有支持该种深度链接的 Client 对象，取其中状态为空闲的一个返回其 ID，若无空闲，则等待 3s 后重新尝试，三次尝试失败后返回查找失败；当有虚拟机变为空闲时，根据 ID 查找 clients 获得对应 Client 对象，将其状态置为空闲。

3.4.4 状态同步模块

该模块主要是部署了一个用于用户手机与安卓虚拟机之间同步的工具。该工具是基于 STF 实现的，通过插入 JS 代码来跳过 STF 的登录界面，使用户手机可以根据虚拟机 ID 直接访问 STF 客户端控制某台虚拟机的页面。在服务器上安装搭建好环境并启动 STF 服务后，在浏览器中就可以通过服务器的 7100 端口（默认端口）访问 STF 客户端。在手机端的浏览器访问 STF 客户端控制某台虚拟机的页面时，仅会显示虚拟机的屏幕，因此可以通过 WebView 访问 STF 来实现用户手机与安卓虚拟机之间的状态同步。

4 实验验证与性能分析

基于实现的深度链接云端执行框架，本文选取典型交互应用，在 WIFI 网络环境下，验证了深度链接执行结果的正确性，并且测量了云端执行的运行开销。为此，本文设计并进行了如下实验。

4.1 正确性验证及执行效果展示

在正确性验证实验中，本文采取如下方案：使用不同的参数调用函数来执行不同类型的深度链接，交替使用正确和错误的 Token，其余参数取决于深度链接的目标页面，在得到交互界面上进一步的操作完成后，点击悬浮按钮关闭交互界

图 6 京东交互界面截图

面。我们共测试了六个应用中八种深度链接的执行，即表 1 的第一列中的八种。

实验结果表明，在单线程线性操作下，云端执行的结果正确。当 Token 正确时，可以正确执行深度链接，打开目标页面，并且可以进行交互操作，点击悬浮按钮后也可以正确关闭交互界面并结束深度链接执行。

接下来为正确执行的效果展示[1]。以执行京东应用搜索页面的深度链接为例，将京东的包名，搜索页面的页面名和搜索关键字作为参数调用 SDK 执行深度链接。在一定的延迟之后，就会获得如图 6 所示的交互界面，可以在该界面上进行进一步的操作。在操作完成后，点击下方的悬浮按钮可以关闭交互界面，完成深度链接的执行。

4.2 运行开销评估

4.2.1 实验设计

对于本文所实现的云端执行框架，其运行开销的测量指标比较多，包括延迟，流量消耗，内存消耗，电量消耗等。但是深度链接本身最大的优点即为方便快捷，因此云端执行的延迟是其中比较重要的指标；而在执行过程中，由于用户手机需要通过视频流的方式与云端虚拟机保持状态同步，可能带来额外的流量开销。因此我们对于延迟与流量消耗这两个指标进行测量评估。在实验中，本文采取如下方案：对于延迟的测量，在函数调用前及调用完成后的位置分别获取系统时间，取其差值作为执行的延迟；对于流量消耗的测量，在若干次调用某一功能函数后，使用 tcpdump 抓包并导出，使用 wireshark 打开抓包文件并过滤，然后统计流量并计算平均每次调用消耗的流量。需要注意的是，对于执行深度链接这一操作，从开始调用执行深度链接，到关闭交互界面完成后才算是一次完整的执行，因此对于这两部分的执行效果放在一起统计，即深度链接执行的延迟为上述两部分延迟之和，用户在交互界面停留的时间不计算在内。深度链接执行的流量消耗为此过程中消耗的总流量，测试时一旦交互界面打开完成后就立即关闭，以此得到的流量消耗值作为深度链接执行的流量消耗。对于获取 Token 这一操作，我们调用 50 次取其平均；对于执行深度链接，我们调用八种深度链接各 5 次取平均。

4.2.2 实验结果

根据上述实验设计进行实验，将得到的数据结果统计后可得到获取 Token 的执行延迟为 70ms，流量消耗为 0.87KB；执行深度链接的延迟及流量消耗如表 1 所示：

根据上述实验结果可以发现，获取 Token 的执行延迟很低，不到 0.1s，流量消耗也很低，不到 1K。而执行深度链接的延迟较高，平均达到 14s 以上，但如表 1 所示，总延迟中 83%为 STF 加载的延迟，由于 STF 冷启动及网络状况等原因，这部分延迟较高，我们的框架的延迟仅有 2.5s 左右；而执行深度链接的平均流量消耗则达到了 4M，虽然比较高，但是相比于这些应用的 APK 大小，可以看到还是有显著的降低的。从 STF 的高加载延迟可以看出，目前深度链接的云端执行效果还存在一定的提升空间。而随着用户在交互界面上停留时间的增加，长时间维持用户与云端虚拟机之间的视频流，云端执行的流量消耗会继续上升。

5 相关工作

5.1 深度链接

随着移动智能终端规模的扩大以及 APP 数量的增长，移动深度链接实现技术的优化与改进变得越来越重要。最近 Google，百度，Facebook 等公司都开始大力推广移动平台上的深度链接这一概念，各自推出了 App Indexing，App Link，App Links 等相关技术[1]。Yun Ma 等人的工作显示，在安卓平台上，深度链接已经变得越来越普及，热门应用中支持深度链接的比例低，这是因为为了支持深度链接所需要的代码量过大，对于开发人员来说，无法通过人工开发的方式达到较

[1] 访问 http://list.youku.com/albumlist/show/id_50489023.html 可以观看演示效果视频

表 1　执行深度链接的测试结果

深度链接种类（APP 包名+目标页面名）	总延迟（ms）	STF 加载延迟（ms）	流量消耗（KB）	APK 大小（MB）
com.jingdong.app.mall+Search	10,257	8,218	2,465	60.11
com.jingdong.app.mall+Product	11,769	9,317	3,210	60.11
com.sdu.didi.psnger+KuaicheHailing	15,279	12,682	4,630	25.41
com.yongche.android+CarHailing	16,020	13,650	4,773	30.01
com.sankuai.meituan+Search	16,078	13,497	4,335	31.97
com.sankuai.meituan+Restaurant	16,022	13,713	4,569	31.97
me.ele+Search	15,160	12,474	4,669	19.84
ctrip.android.view+Search	18,159	15,399	5,304	51.15
平均值	14,718	12,369	4,053	38.82

高的覆盖率[2]。对于这个问题，Tanzirul Azim 等人提出了 uLink 这一技术，跟踪并记录应用页面的数据和 UI 事件，在调用深度链接时利用这些数据快速准确的生成目标页面，由此可以减少开发者的工作量，提高覆盖率，并在链接失效时给出准确反馈信息[7]。Yun Ma 等人则提出了 DroidLink 这一技术来自动生成深度链接，通过静态和动态分析，记录到达应用中每个页面的跳转过程，并根据这些信息为应用自动生成深度链接，由此最大程度上减少了开发者的工作量[8]。

综上所述，随着深度链接技术的发展以及各种新技术的提出，移动深度链接将会变得越来越普及，而且应用的安装覆盖率呈现长尾分布，深度链接目标应用未安装的情况会越来越多。基于本文的工作，在目标应用未安装的情况下也可以达到与直接执行一致的用户体验，使得深度链接更加容易执行，也更加容易被使用。

5.2　远程状态同步

本文中所实现的云端执行在云端的虚拟机执行完深度链接后，需要用户手机与云端虚拟机保持状态同步，以便进一步的操作。具有类似功能的工具很多，但都存在一定不足，无法满足本文的需要。

例如工业界有 Mobizen[9]等工具，但是这些工具只能运行在电脑上，即电脑控制手机，无法做到手机与手机的远程状态同步。还有各大手机开发商开发的远程协助工具，如小米远程协助、华为亲情关怀等，但是这些工具都需要安装应用才能使用，有些甚至无法达到实时的状态同步。学术界有一些工具也能达到这样的效果，例如 SVMP[10]，这是一个基于 AOSP 实现的云端虚拟机管理部署工具，可以在手机上创建、连接并操作云端虚拟机。其优点在于安全性高，同步性强，但是同样存在手机必须安装客户端才可以使用的问题。

而本文选用的 STF 是一个 Web 端工具，可以不在用户手机上安装额外的应用，直接在 webview 中就可以访问 STF 服务，做到与虚拟机的状态同步。

结束语　随着移动应用市场发展，用户对安卓应用中深度链接执行方式的需求发生了变化，已有的执行方式无法满足用户的新需求。基于此，本文提出了一种安卓应用深度链接的云端执行框架，在云端安卓虚拟机中执行深度链接，手机与云端虚拟机进行通信，进而在未安装应用时也能达到与直接执行一致的用户体验。针对提出的框架，本文基于已有的工具，设计并实现了一组 SDK 支持这种框架，并且设计实验验证了本文实现的云端执行框架的正确性，测量了其执行效果，包括延迟和流量消耗。

目前云端执行框架还存在一些不足，未来可以从以下几个方面进行改进：进一步改善 STF 框架，减少流量消耗及加载延迟；改进服务器在压力负载下的调度方法；优化客户端与服务器的代码以提高效率；在服务器端增加对用户数据和应用数据的维护。同时还需要继续进行试验，验证云端执行的效果，在系统进一步优化后，需要验证执行过程的稳定性，验证在客户端多线程并行的情况下能否顺利完成调用。需要进一步确定执行效果的度量机制，并在此度量机制下，进行

大量实验以比较云端执行与延迟执行的性能差异，并且增加对用户交互操作的延迟及流量的测试。

参考文献

[1] OUYANG C. The past life of mobile DeepLink [Z/OL].(2016) .http://mp.weixin.qq.com/s?__biz=MzAxMTMxNjUxMA==&mid=404578715&idx=1&sn=ec2c0d3ba680a16efebe79bb3d5a5055&mpshare=1&scene=1&srcid=04100xIPg9CXONjYaFdLDFjG#rd.

欧阳辰. 移动 DeepLink 的前生今世[Z/OL]. (2016). http://mp.weixin.qq.com/s?__biz=MzAxMTMxNjUxMA==&mid=404578715&idx=1&sn=ec2c0d3ba680a16efebe79bb3d5a5055&mpshare=1&scene=1&srcid=04100xIPg9CXONjYaFdLDFjG#rd.

[2] MA Y, LIU X, HU Z, et al. Aladdin: automating release of Android deep links to in-app content[C]// Proceedings of the 39th International Conference on Software Engineering Companion. IEEE Press, 2017:139-140.

[3] LI H, AI W, LIU X, et al. Voting with Their Feet: Inferring User Preferences from App Management Activities[C]// Proceedings of the 25th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2016: 1351-1362.

[4] KINNUNEN S, BRUNNER G. STF[CP/OL].https://openstf.io/.

[5] HITCHENS R. Java NIO[M]. USA: O'Reilly Media, 2002.

[6] JIN Z, LI W. Design and implementation of HTTP client based on Netty[J]. Telecom Engineering Technics and Standardization, 2014(1):84-88.

金志国, 李炜. 基于 Netty 的 HTTP 客户端的设计与实现[J]. 电信工程技术与标准化, 2014(1):84-88.

[7] AZIM T, RIVA O, NATH S. uLink:Enabling User-Defined Deep Linking to App Content[C]// Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2016:305-318.

[8] MA Y, LIU X, DU R, et al. DroidLink: Automated Generation of Deep Links for Android Apps[J]. arXiv preprint arXiv:1605.06928, 2016.

[9] Rsupport. Mobizen[CP/OL].https://www.mobizen.com/.

[10] KEPPLER D. Virtual Smart Phones in the Cloud [CP/OL].http://svmp.github.io/.

[11] COMBS G, RAMIREZ G, BOTTOM T. Wireshark[CP/OL].https://www.wireshark.org/docs/.

[12] JBoss. Netty project[CP/OL].(2012-4-1) .http://netty.io/.

[13] JACOBSON V, LERES C, MCCANNE S. TCP DUMP[CP/OL].http://www.tcpdump.org/tcpdump_man.html#lbAI.

[14] google-gson[CP/OL].http://code.google.com/p/google-gson/.

[15] Introducing JSON[CP/OL].http://json.org/json-zh.html.

作者：刘譞哲。联系电话：010-62757077。邮箱：liuxuanzhe@pku.edu.cn