

FUSION:

A Unified Application Model for Virtual Mobile Infrastructure

Student: Chieh-Min Wang
Advisor: Dr. Yu-Sung Wu

2016/7/25

Outline

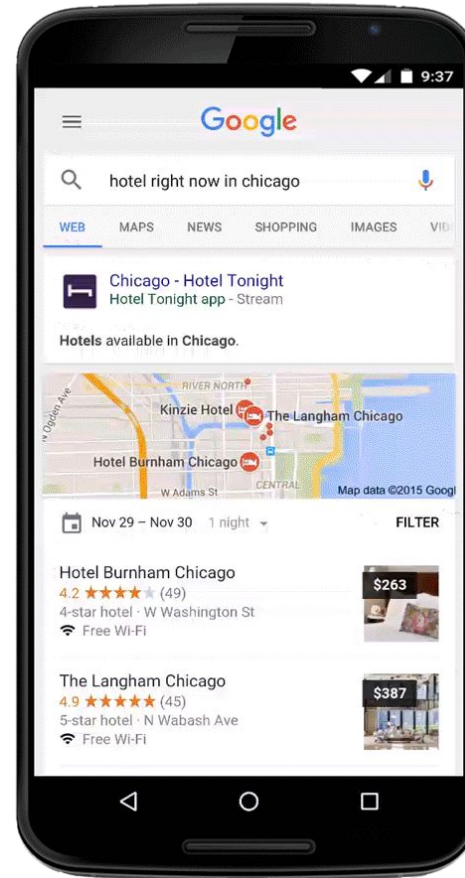
- Introduction
- Background
- Design
 - FUSION Architecture
 - FUSION Android VM
 - Intercepting IPC Events
 - IPC Event Redirection
 - Resource Synchronization
- Implementation
 - Refactoring SVMF for Android x86
 - WebRTC and WebSocket communication links
 - Intent Serialization and Data Format
 - Intent Hooking and Replaying
 - Intents with File Access Requests

Outline

- Evaluation
 - DEMO
 - Responsiveness
 - Performance Overhead
 - Case Studies
 - Limitations
- Related Work
- Conclusion
- Q&A

Introduction

- VMI and app streaming are becoming more popular today
 - Google acquired Agawi .Inc and had launched the app streaming test
 - VMFIVE launched AdPlay service



Introduction

- VMI and app streaming environment are mostly isolated from the local phone environment
- FUSION is designed to enable the interactions between remote VMI environment and the local environment
- The remote VMI environment and the local device environment can be regarded as a unified environment

Background

- Virtual mobile infrastructure(VMI) is a platform hosting virtual mobile operating system
 - The virtual machine's screen content is passed to the user's thin client
- Application streaming is a technique built on top of the VMI
 - Only passing specific apps' screen content to the user's thin client
 - Users feel the remote apps are just like regular apps installed on the local device
- Current solutions do not allow remote apps to interact with local apps
- FUSION bridges the gap between the VMI environment and the local device
 - Bi-directional IPCs
 - Loosely synchronized file system

Background: Android IPC events

- The most common and standard IPC event: Intent
 - Ask other apps to do some specific tasks for the users
- Intent structure
 - action -- The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_SEND, etc.
 - data -- The data to operate on, such as a file stored on the device, expressed as a Uri.
 - component -- Specifies an explicit name of a component class to use for the intent.

Background: Android IPC events

- Explicit Intent

- A way for an application to launch various internal activities
- Specify a component via *setComponent(ComponentName)* and *setClass(Context, Class)*

```
Intent intent = new Intent(this, ActivityABC.class);  
i.putExtra("Value", "This value for ActivityABC");  
startActivity(intent);
```

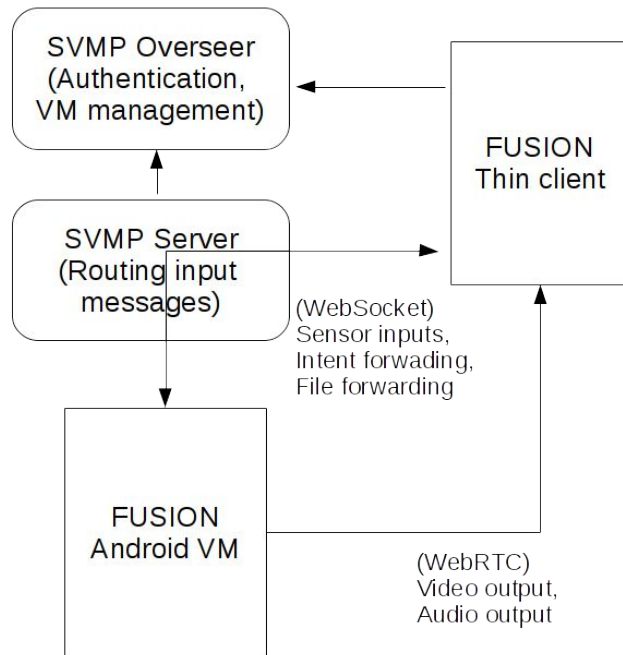
- Implicit Intent

- Do not specify a component; Commonly used for communication between apps
- Include enough information for the system to determine which component to run for the intent (via *setAction(String)*, *setData(Uri)*, *setType(String)* and *setDataAndType(Uri, String)*)

```
Intent intent = new Intent(Intent.ACTION_VIEW,  
                           Uri.parse("http://www.example.com"));  
startActivity(intent);
```

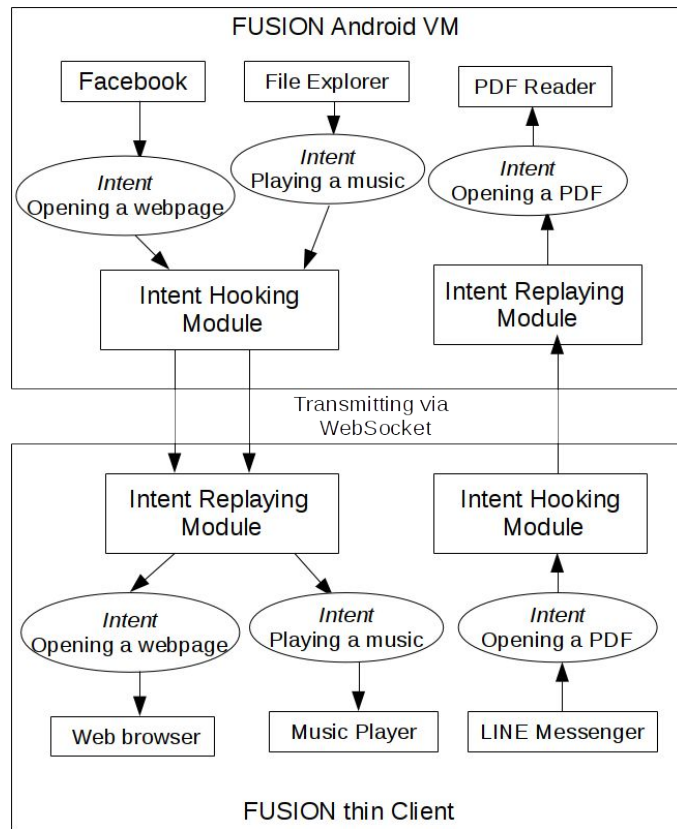

Design - FUSION Architecture

- FUSION implementation is based on SVMP project
 - SVMP project has been ceased since two years ago
- SVMP overseer
 - User login and authentication
- SVMP server
 - Routing input messages
- FUSION thin client
- FUSION Android virtual machine



Design - FUSION Architecture

- Intent hooking module and intent replaying module on both FUSION Android VM and FUSION thin client
- Intent hooking module
 - Intercept Android intent IPC events
 - Forward captured intents to the remote peer
- Intent replaying module
 - Replay intents

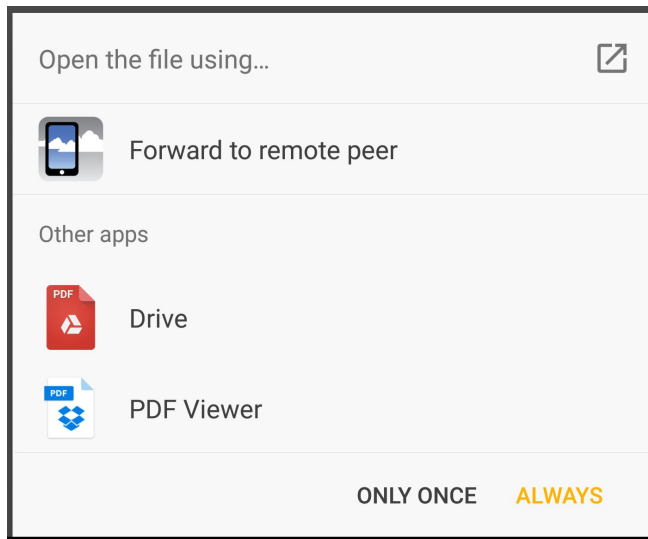


Design - FUSION Android VM

- SVMP Android VM is based on AOSP project
 - OpenGL does not work
 - Lacks GPU drivers for GPU passthrough
- FUSION Andorid VM is a port of SVMP from AOSP to Android x86
 - Using the drm_gralloc HAL implementation
 - Full support of GPU hardware that are compatible with standard x86 Linux
 - Newer Kernel support

Design - Intercepting IPC Events

- An intent hooking activity registering intent-filters that can intercept all intents
- ActivityManagerService will look for compatible activities installed on the system
- A prompt dialog when multiple matching activities are found



Design - IPC Event Redirection

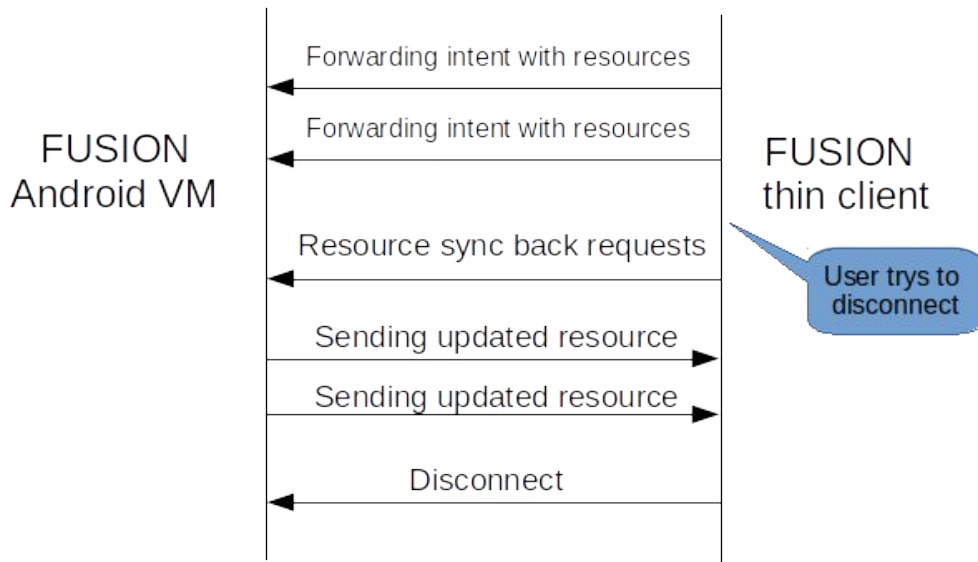
- Intent Hooking Module is responsible for hooking and forwarding intents
- Implicit Intent uses data field to specify resource used by Intent
 - Data is a Uri which conforms to RFC 2396
- FUSION classifies intents into two types by the uri scheme
 - Uri with remote resources (ex: `http://senselab.tw`)
 - Uri with local resources (ex: `file:///sdcard/example.txt`)

Design - IPC Event Redirection

- Uri scheme for remote resource: http, https and ftp...etc
 - Simply serializes and transmits the intents with all its fields
- Uri scheme for local resource: file
 - Serialize the local resource into a byte format
 - Pack the serialized resource into intent message
 - De-serialize the resource and store it into the remote environment
 - Modify uri path to match the file system structure of the remote environment
 - Some directory may not exist on the remote environment

Design - Resource Synchronization

- Ensure the resource modifications can be seen on both sides.
- FUSION records the resource meta data for every forwarded resource
- Synchronize resource back when user tries to disconnect



Implementation - Refactoring SVMP for Android x86

- Port the user-mode SVMP daemon to Android x86
 - Switch to Trebuchet for launcher setting
- Port SVMP's AOSP framework changes to Android x86
 - Ex: Interception of notifications in NotificationManager
 - Android version from 4.4.4_r2 to 4.4.4_r2.0.1
- Port SVMP HAL libraries
 - libaudio and libsensor
 - Set ro.hardware.audio.primary and ro.hardware.board to corresponding value
- LOC changes
 - device_sense_svmp: 532 loc
 - android_external_svmp_eventserver: 494 loc
 - Other framework changes: 846 loc

Implementation - Communication links

- Screen content and audio output are transmitted via WebRTC
- Sensor data, location updates and intent/resource message are transmitted via WebSocket
 - Serialized to ProtocolBuffer format
 - Use AutoBahn websocket implementation
 - FUSION modifies AutoBahn WebSocket implementation to transmit big file over WebSocket
- NAT traversal
 - Bind the host machine with a public IP (Full Cone NAT for Android VM)
 - Use Google's public STUN server

Implementation - Intent serialization

- Extend existing ProtocolBuffer interface
- Each field is one-to-one correspondent to the Android Intent class except the file field
- file field is used to store the serialized local resource

```
1  message Intent {  
2      required IntentAction action = 1;  
3      repeated Tuple extras = 2;  
4      optional string data = 3;  
5      optional string type = 6;  
6      optional File file = 7;  
7      repeated int32 flags = 4;  
8      repeated string categories = 5;  
9      message Tuple {  
10         required string key = 1;  
11         required string value = 2;  
12     }  
13 }
```

Implementation - Intent Hooking

- Define multiple intent-filters to hook all kinds of intents
- There are many action type: ACTION_VIEW, ACTION_SEND, each has its own intent-filter
- Custom action type is not supported by FUSION

```
1 <intent-filter>
2   <action android:name="android.intent.action.VIEW" />
3   <category android:name="android.intent.category.DEFAULT" />
4   <category android:name="android.intent.category.BROWSABLE" />
5   <data android:host="*" android:scheme="http"/>
6   <data android:host="*" android:scheme="https"/>
7   <data android:mimeType="application/*" />
8   <data android:mimeType="audio/*" />
9   <data android:mimeType="image/*" />
10  <data android:mimeType="message/*" />
11  <data android:mimeType="multipart/*" />
12  <data android:mimeType="text/*" />
13  <data android:mimeType="video/*" />
14 </intent-filter>
```

Implementation - Intent Replaying

- Simply reconstruct intent with message received
- Adjust Uri path if needed
 - Ex: /sdcard/Documents/haha/pokemon.jpg
 - /sdcard/Documents/haha/ directory may not exist in the remote environment

```
1 SVMPProtocol.Intent intentRequest = request getIntent();
2 Intent intent = new Intent(Intent.ACTION_VIEW);
3 intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
4 intent.setData(Uri.parse(intentRequest.getData()));
5 baseServer.getContext().startActivity(intent);
```

Implementation - Processing URIs

- Classify the type of intent by its scheme

```
1 Uri data = intent.getData();
2 SVMPProtocol.File.Builder f = null;
3 if(data.getScheme().equals("file")) {
4     f = SVMPProtocol.File.newBuilder();
5     f.setFilename(data.getLastPathSegment());
6     f.setData(getByteArray(data));
7 }
8 SVMPProtocol.Intent.Builder intentBuilder = SVMPProtocol.Intent.newBuilder();
9 ...
10 if(file != null) {
11     intentBuilder.setFile(file);
12 }
13 ...
```

Implementation - Files Serialization

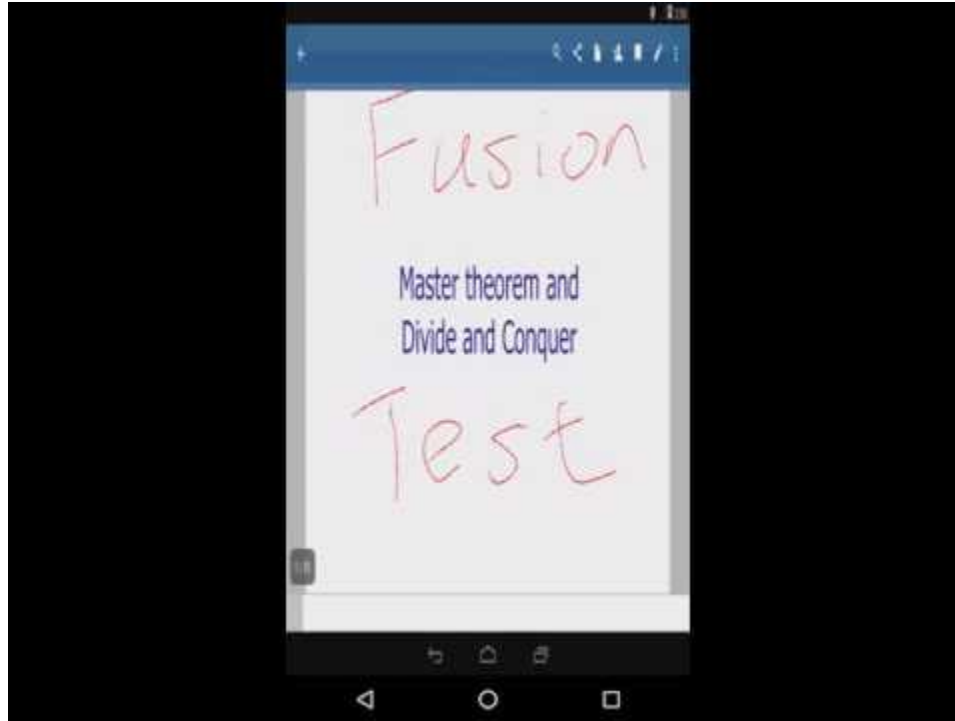
- Serialize to ProtocolBuffer's ByteString type(Java binding for byte array)

```
1 private ByteString getByteString(Uri uri) {
2     try {
3         InputStream iStream = mContext.getContentResolver().openInputStream(uri);
4         byte[] inputData = getBytes(iStream);
5         return ByteString.copyFrom(inputData);
6     } catch (Exception e) {}
7     return null;
8 }
9 private byte[] getBytes(InputStream inputStream) throws Exception {
10    ByteArrayOutputStream byteBuffer = new ByteArrayOutputStream();
11    int bufferSize = 1024;
12    byte[] buffer = new byte[bufferSize];
13    int len = 0;
14    while ((len = inputStream.read(buffer)) != -1) {
15        byteBuffer.write(buffer, 0, len);
16    }
17    return byteBuffer.toByteArray();
18 }
```

Implementation - Resources Synchronization


- forwardedFiles and waitingFiles list in the Intent Hooking Module
 - forwardedFiles: record all resource which is forwarded to the remote peer
 - waitingFiles: record the resource that have not been synchronize back
- Maintain the two lists to handle resource forwarding and sync back simultaneously
- Resource synchronization stage will be triggered when the user tries to disconnect from the remote peer
 - Send file sync back requests for each resource in forwardedFiles list
 - Move all content from forwardedFiles into waitingFiles
 - waitingFiles list is a HashSet data structure in Java as the order of files may not be in the same order as we send the file sync back request

Evaluation - DEMO



Evaluation

- Intel i7-4770 8-core processor, 32GB memory
- Each of the Android VMs is equipped with 4 VCPUs and 4G RAM
- Use QEMU/KVM to emulate peripheral devices and virtualize the CPU



```
1 #!/bin/bash
2 qemu-system-x86_64 --enable-kvm \
3 -hda ./FUSION_android.qcow2 -serial stdio \
4 -m 4G -smp 4 -net nic,vlan=0 \
5 -net user,hostfwd=tcp:127.0.0.1:8001-:8001,hostfwd=tcp:127.0.0.1:5555-:5555 \
6 -soundhw all -vga std
```

Evaluation - Responsiveness

- Pure intent forwarding(no file forwarding involved)
- Intent forwarding with resources forwarding of different file size
- Log the timestamps at the hooking points in the intent hooking module and the receiving points in the intent replaying module
- Use PureVPN to emulate the deployment of FUSION on wide-area-network environment

Evaluation FUSION induced delay

- Open a links in Facebook with remote browser
- Compare with RTT detected by ping tool

	FUSION transmission time (ms)	Ping RTT (ms)
NCTU campus network	637	4
ChungHwa Telecom 4G LTE	1028	79
VPN through Netherlands	1920	722
VPN through United State(New York)	1537	765

Evaluation - File synchronization delay

- Generate dummy files using dd

```
1 #!/bin/bash
2 SIZE=$1
3 dd if=/dev/zero of=dummy.txt bs=$SIZE count=1
```

- Evaluate using NCTU campus network(i.e the server and client is located in the same network)

Evaluation - File synchronization delay

- WebSocket is not designed for transmitting large files
- Most files storing on mobile devices are not big
 - most mp3 music files or jpeg photo files are less than 5MB

	Synchronization Time (ms)
1 MB	2086
5 MB	3649
10 MB	7766
20 MB	12204
40 MB	21943

Evaluation - Performance Overhead

- Benchmark: Geekbench 3
- Performance overhead is less than 1%

	Single-Core Score	Multi-Core Score
Vanilla Android x86	2648	7127
SVMP Android	2736	7364
FUSION Android without client connection	2665	7135
FUSION Android with client connection	2575	7050
Nexus 6 (Android 6.0)	1042	3050

Evaluation - Case Studies: Opening link



Open with FUSION Client

JUST ONCE ALWAYS

Use a different app



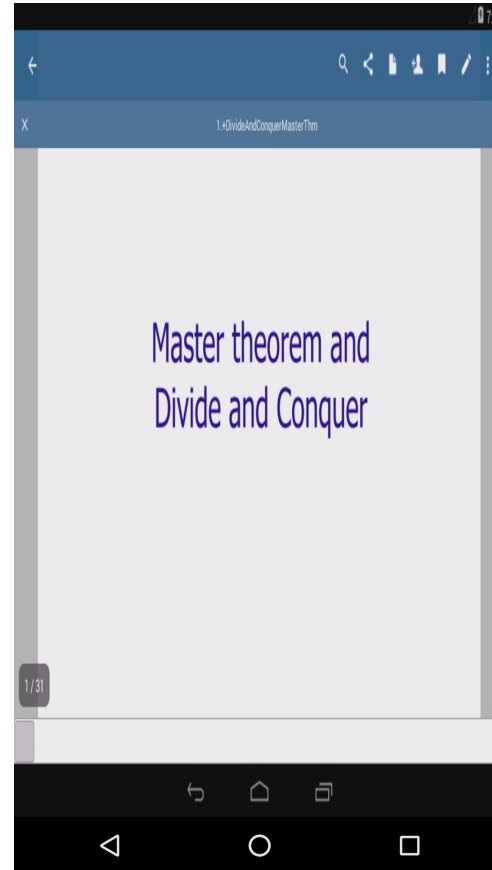
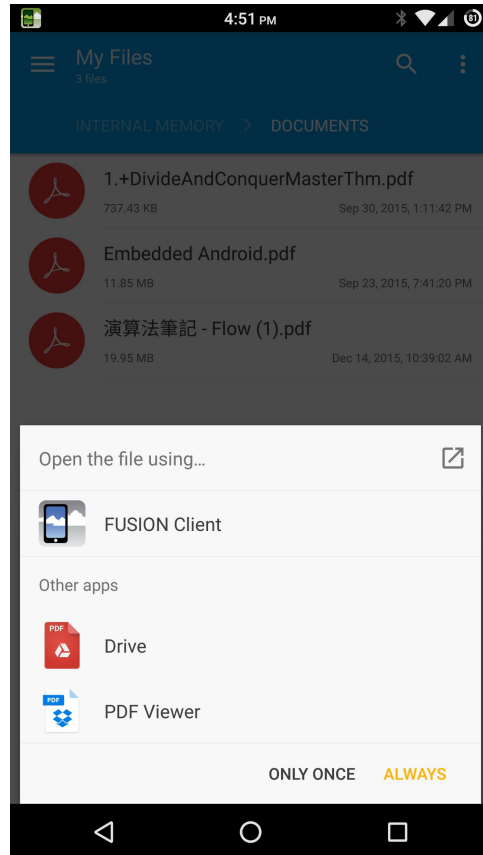
Chrome



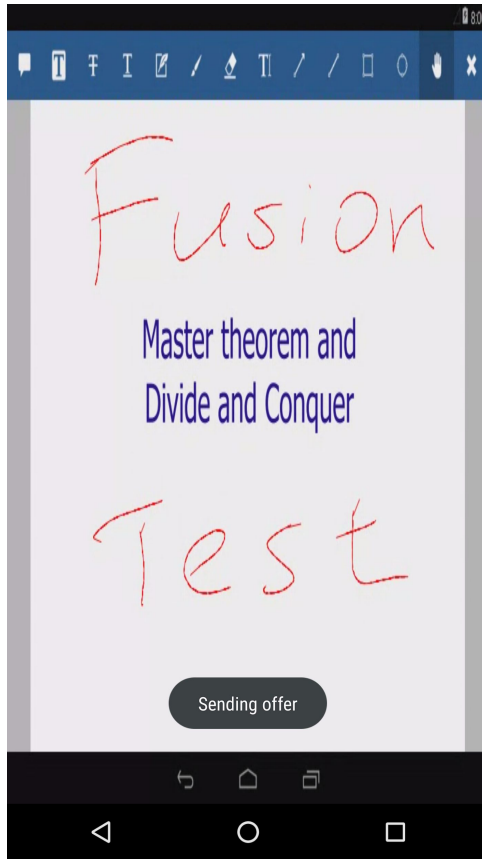
Puffin



Evaluation - Case Studies: Viewing PDF



Evaluation - Case Studies: Viewing PDF



Fusion

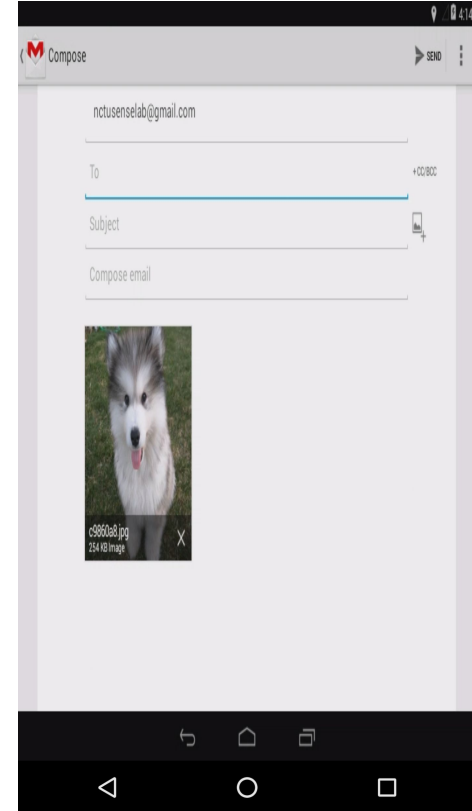
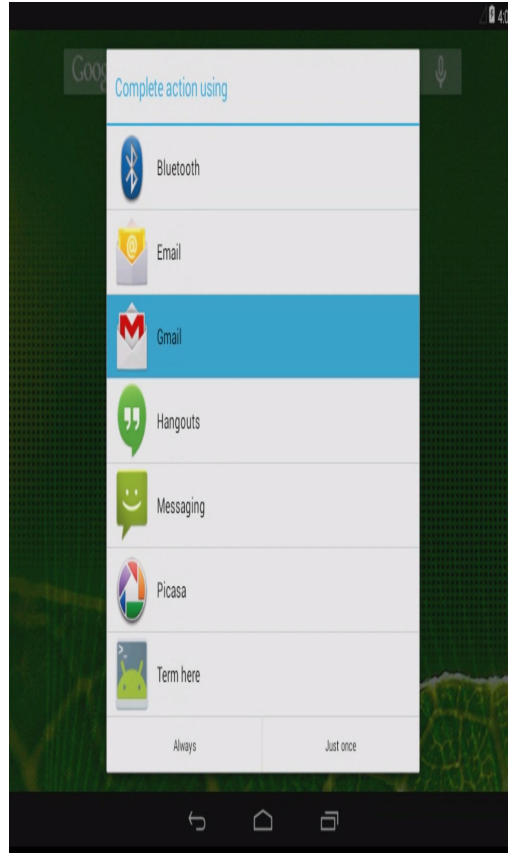
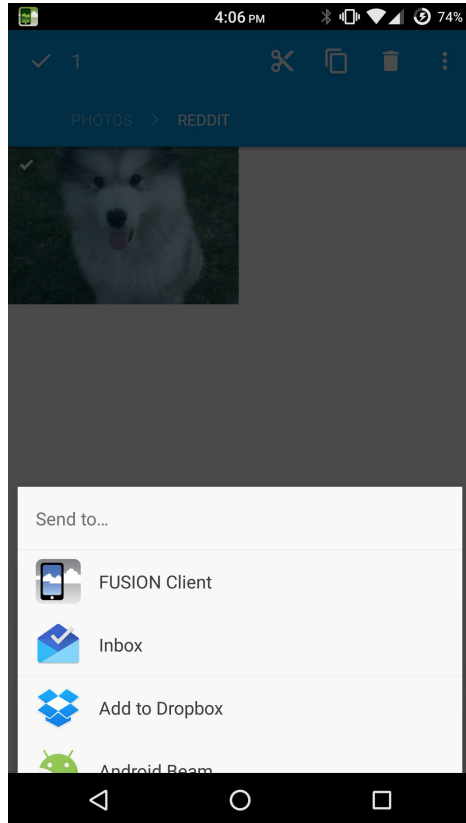
Master theorem and
Divide and Conquer

Test

The divide-and-conquer design paradigm

- 1. Divide** the problem (instance) into subproblems.
- 2. Conquer** the subproblems by solving them recursively.
- 3. Combine** subproblem solutions.

Evaluation - Case Studies: Sharing Files



Evaluation - Limitations

- For app using `startActivityForResult` will not work
 - Require injecting the result into the `ActivityManagerService`
 - Framework modification is not allowed in VMI and app streaming scenario
- Custom action type defined by app is not supported by FUSION
 - Ex: Facebook and Facebook Messenger may use intents with custom action type

Related Work

- Remote desktop has been researched and developed for a long time
 - Microsoft RDP, VNC, SPICE...etc
 - Provide simple integration: clipboard sharing and file sharing
- US patent "method of providing a remote desktop session with the same look and feel as a local desktop"
 - Makes the remote app with the same look and feel as the windows of local app
 - Transferring the values of attributes of the local desktop to the server nodes
- US patent "methods and systems for incorporating remote windows from disparate remote desktop environments into a local desktop environment"
 - makes the remote servers draw directly on the local windows
 - one channel conveys graphical data
 - another channel conveys window attribute data

Related Work

- All of the previous studies lack comprehensive system integration including bi-direction IPC event transmission and file system synchronization as provided by FUSION
- FUSION is novel because it makes applications in the two environments can communicate with each other directly

Conclusion

- VMI and app streaming provide a more convenient and scalable way for using mobile devices and apps
- Existing solutions do not provide interaction between remote VM and local device
- FUSION enables IPC interaction and bridges the remote VM and local device as an unified environment
- FUSION provides a loosely synchronized file system to bridge the two environments
- Intent and resource serialization and transmission are efficient
- The performance impact of FUSION on the VM is less than 1%

Q&A