Discussions on the next generation OS

OS Design for the Future of a Connected World

Xiao-Feng Li
xiaofeng.li@gmail.com
January 2020
Agenda

- OS from individual to group
- OS from application to service
- OS from I/O to sensing
- Survey of the industry
OS design is evolving

- Trend: from hardware enabling towards user services.
OS components in the highest level

- A consumer OS is comprised of three components
  - Resource management (for vendors)
  - APIs (for app developers)
  - Interfaces (for users)

- Trend: most innovations in recent years are in app development. Methodology of hardware enabling and user interactions changes slowly.
What decides a consumer OS?

- Why we call it Android
  - Not u-boot, little kernel, Linux, super app, although all the rest are OSes too
  - Because the primary APIs are provided by Android
  - Most apps rely on the services provided by Android framework
  - OS is decided by the primary APIs
OS service is changing

- Trend: from resource management towards daily assistant
Levels of service APIs

- Diff services use diff protocols at diff levels, because of the design legacy from ground up accumulatively.

<table>
<thead>
<tr>
<th>Levels</th>
<th>Data source</th>
<th>Functionalities</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>File system</td>
<td>Hardware abstraction</td>
<td>System call</td>
</tr>
<tr>
<td>Inter-process</td>
<td>Data server</td>
<td>App features</td>
<td>IPC</td>
</tr>
<tr>
<td>Local</td>
<td>Device</td>
<td>P2P/AV/Sensor</td>
<td>WiFi/BT/NFC</td>
</tr>
<tr>
<td>Remote</td>
<td>Internet</td>
<td>Cloud services</td>
<td>HTTP</td>
</tr>
</tbody>
</table>

*Only examples, not a full list.
One protocol for all services

*Android status as an example.

<table>
<thead>
<tr>
<th>Process</th>
<th>Data source</th>
<th>Functionalties</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter-process</td>
<td>File system</td>
<td>Hardware abstraction</td>
<td>HIDL</td>
</tr>
<tr>
<td>Local</td>
<td>Data server</td>
<td>App features</td>
<td>AIDL</td>
</tr>
<tr>
<td>Remote</td>
<td>Device</td>
<td>P2P/AV/Sensor</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>Internet</td>
<td>Cloud services</td>
<td>Google APIs</td>
</tr>
</tbody>
</table>

- Android is evolving close to a consistent protocol through Binder, except for local services
Local area protocols

- Google has Nearby/Wear/Auto/Cast protocols, specifically for different devices.
- They are not general enough to various devices. A unified solution is,
  - Extend Binder to cross-devices services
  - With multi-channel Audio/Video streaming
  - With Nearby for discovery and connection
  - With multi-display for other scenarios
Another view of OS architecture

Android’s “microkernel” design
Agenda

❖ OS from individual to group
❖ OS from application to service
❖ OS from I/O to sensing
❖ Survey of the industry
Service-oriented OS

- Modern OS is around services, instead of hardware
  - Service protocol: binder IPC
  - Registration/discovery: package manager
  - Lifetime/session: activity manager
  - User interface: GUI+touch
  - External events: notification manager
  - Data service: content provider
# OS: From individual to group

*Android status as an example.*

<table>
<thead>
<tr>
<th>OS</th>
<th>Individual</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service protocol</td>
<td>IPC</td>
<td>Wear/Cast/Auto/…</td>
</tr>
<tr>
<td>Service discovery</td>
<td>package manager</td>
<td>Nearby/WiFi</td>
</tr>
<tr>
<td>Service session</td>
<td>activity manager</td>
<td>? (device-specific)</td>
</tr>
<tr>
<td>Service data</td>
<td>content provider</td>
<td>? (device-specific)</td>
</tr>
<tr>
<td>External events</td>
<td>notification manager</td>
<td>? (device-specific)</td>
</tr>
<tr>
<td>Service interface</td>
<td>GUI/touch</td>
<td>? (device-specific)</td>
</tr>
</tbody>
</table>

- Mostly other devices act as accessories to phone. Very little about p2p services or group services.
Build a group OS

<table>
<thead>
<tr>
<th>OS</th>
<th>Individual</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service protocol</td>
<td>IPC</td>
<td>RPC</td>
</tr>
<tr>
<td>Service discovery</td>
<td>package manager</td>
<td>Nearby manager</td>
</tr>
<tr>
<td>Service session</td>
<td>activity manager</td>
<td>session manager</td>
</tr>
<tr>
<td>Service data</td>
<td>content provider</td>
<td>distributed data</td>
</tr>
<tr>
<td>External events</td>
<td>notification manager</td>
<td>distributed events</td>
</tr>
<tr>
<td>Service interface</td>
<td>GUI/touch</td>
<td>voice/gesture</td>
</tr>
<tr>
<td>Meta API</td>
<td>Tasker/Shortcuts</td>
<td>DSL</td>
</tr>
</tbody>
</table>

- Android can be extended for a group OS
Data are services too

- Services today only provide end-to-end integrated data
- Users should regain the control of their own data

<table>
<thead>
<tr>
<th>Data source</th>
<th>Examples</th>
<th>Exposed as services</th>
</tr>
</thead>
<tbody>
<tr>
<td>External service</td>
<td>apps, movies, books</td>
<td>DRM/blockchain</td>
</tr>
<tr>
<td>Self produced</td>
<td>photos, credit cards, passwords, app state</td>
<td>Distributed data</td>
</tr>
<tr>
<td>Runtime data</td>
<td>video stream, sensor data</td>
<td>Exposed with APIs</td>
</tr>
</tbody>
</table>
Services: from local to external

- Information mainly comes from service provider and social network

Cloud

- push/pull
  - E.g., news/search

Group OS

- push/pull
  - E.g., message/follow
External service entry points

❖ Three main entry points
  ❖ Pull: Search (Google, Baidu, …)
  ❖ Push: News (Yahoo, Toutiao, …)
  ❖ Bidirectional: Social (Facebook, Wechat, …)

❖ (Notes: Once dominating one entry, anxious to dominate others. Other services, e.g., maps, etc. are natural extensions.)

❖ OS for next generation should support the models natively
Apps readiness to external services

- Android app is barely external service ready.
- H5 Applet: app ≈ doc ≈ info ≈ social ≠ cloud

<table>
<thead>
<tr>
<th></th>
<th>PWA</th>
<th>H5 Applet</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page searchable</td>
<td>Internet</td>
<td>In super app</td>
<td>Slice in phone App in store</td>
</tr>
<tr>
<td>State recoverable</td>
<td>Partially</td>
<td>Partially</td>
<td>Yes</td>
</tr>
<tr>
<td>Optional install</td>
<td>Yes</td>
<td>Yes</td>
<td>Instant app</td>
</tr>
<tr>
<td>Cross-platform</td>
<td>Yes</td>
<td>Yes</td>
<td>Partially</td>
</tr>
<tr>
<td>Native access</td>
<td>Partially</td>
<td>Partially</td>
<td>Yes</td>
</tr>
<tr>
<td>Composable</td>
<td>No</td>
<td>No</td>
<td>Bundle/Binder</td>
</tr>
<tr>
<td>Distributable</td>
<td>Yes</td>
<td>Yes</td>
<td>Partially</td>
</tr>
</tbody>
</table>
Agenda

❖ OS from individual to group
❖ OS from application to service
❖ OS from I/O to sensing
❖ Survey of the industry
HCI is from I/O to sensing

- User interface becomes more and more natural
Context-aware OS

- OS has to receive inputs from devices around it, and present outputs to suitable devices.
- Inputs are not necessarily immediate info, can be long term profile.
Inputs/outputs are services

- Input - computing - output should be decoupled and distributed

- Application development still follows MVC-class models, and here V is essentially based on I/O services

- OS APIs provide seamlessly access to the decoupled M-V-C service providers
Remote View

- Each has their respective use cases:
  - (Virtual) display mirror
  - Remote frame-buffer / desktop
  - Remote rendering
  - UI client distribution
  - Align the inputs, synchronize the outputs
  - Critical for AV experiences
Service semantics

- Services are mostly binary interfaces
- Not human readable
- Not web searchable
- Not directly map to human activities
- Services should be defined with semantics
- Describe services with human readable tokens
- Services can be connected according to the tokens
DSL as high level APIs

- DSL program can be developed or generated
- Programmable like IFTTT
- Generated from human multi-modal instructions
  - Human instructions → Sequence of semantic tokens → DSL program → OS service operations
- AI engine is mandatory
- DSL program can be converted to common app
  - Managed by app store and is distributable
Abstraction and modularity

- Context-aware OS involves lots of dynamic resources and states
  - Device loads components according to context
  - Different devices loads different components
  - Devices can use different OSes but agree on protocols
- Considerations
  - System update, app migration, API stability, real time, account, security, etc.
Agenda

❖ OS from individual to group
❖ OS from application to service
❖ OS from I/O to sensing
❖ Survey of the industry
Design the next generation OS based on existing one, by enhancing the capabilities of,

- Distributed and connected
- Service-oriented APIs
- Natural interactions