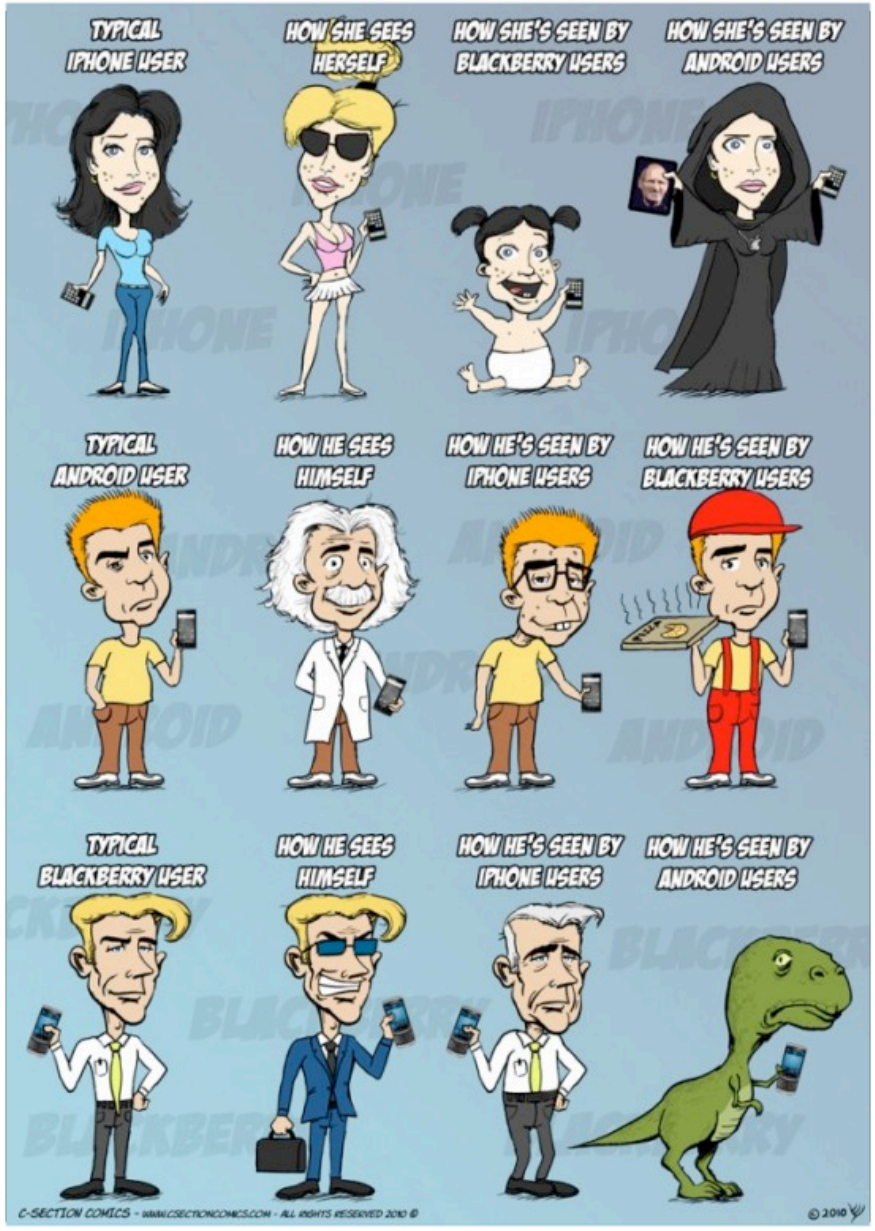
ANDROID OS

CSE120 (FA10)

Xiao Ma
(xiao@xiao-ma.com)


TYPICAL IPHONE USER
HOW SHE SEES HERSELF
HOW SHE'S SEEN BY BLACKBERRY USERS
HOW SHE'S SEEN BY ANDROID USERS
TYPICAL ANDROID USER
HOW HE SEES HIMSELF
HOW HE'S SEEN BY IPHONE USERS
HOW HE'S SEEN BY BLACKBERRY USERS
TYPICAL BLACKBERRY USER
HOW HE SEES HIMSELF
HOW HE'S SEEN BY IPHONE USERS
HOW HE'S SEEN BY ANDROID USERS
C-SECTION COMICS - WWW.CSECTIONCOMICS.COM - ALL RIGHTS RESERVED 2010 ©
© 2010

WHY ANDROID?

the definition of open: "mkdir android ; cd android ; repo init -u git://android.git.kernel.org/platform/manifest.git ; repo sync ; make"

- What does it mean to researchers?
- What does it mean to users?

OUTLINE

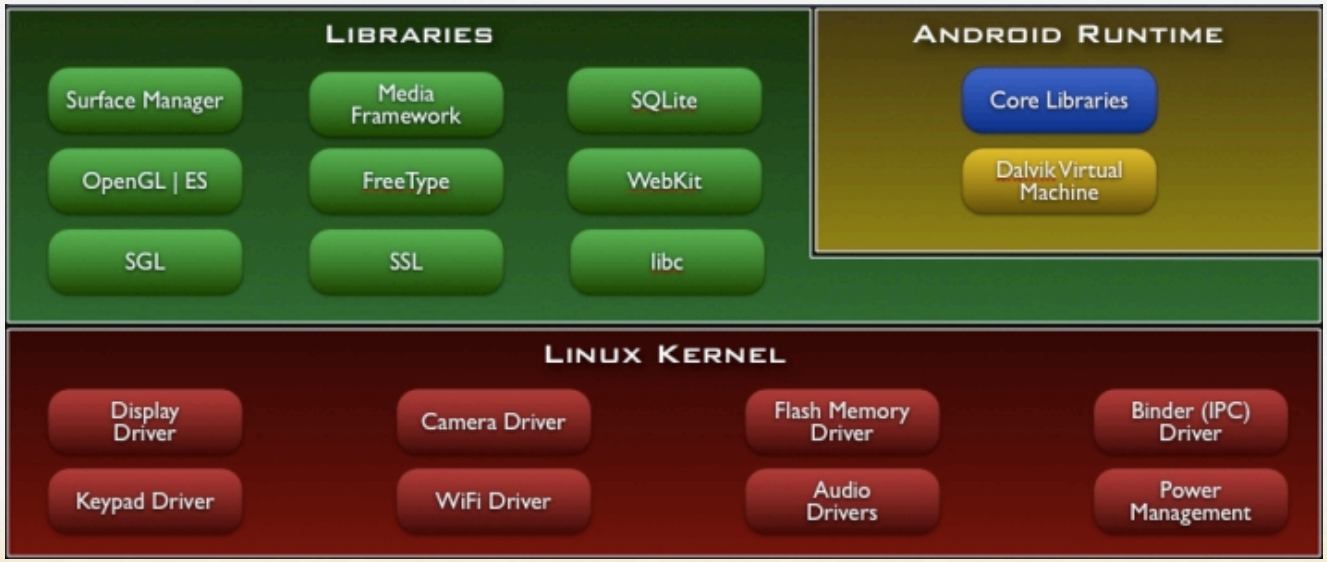
- Android platform architecture
- OS kernel, libraries and devices
- Android programming model
- Delvik Virtual Machine
- Energy efficiency
- How to write efficient code

FIRST THING FIRST

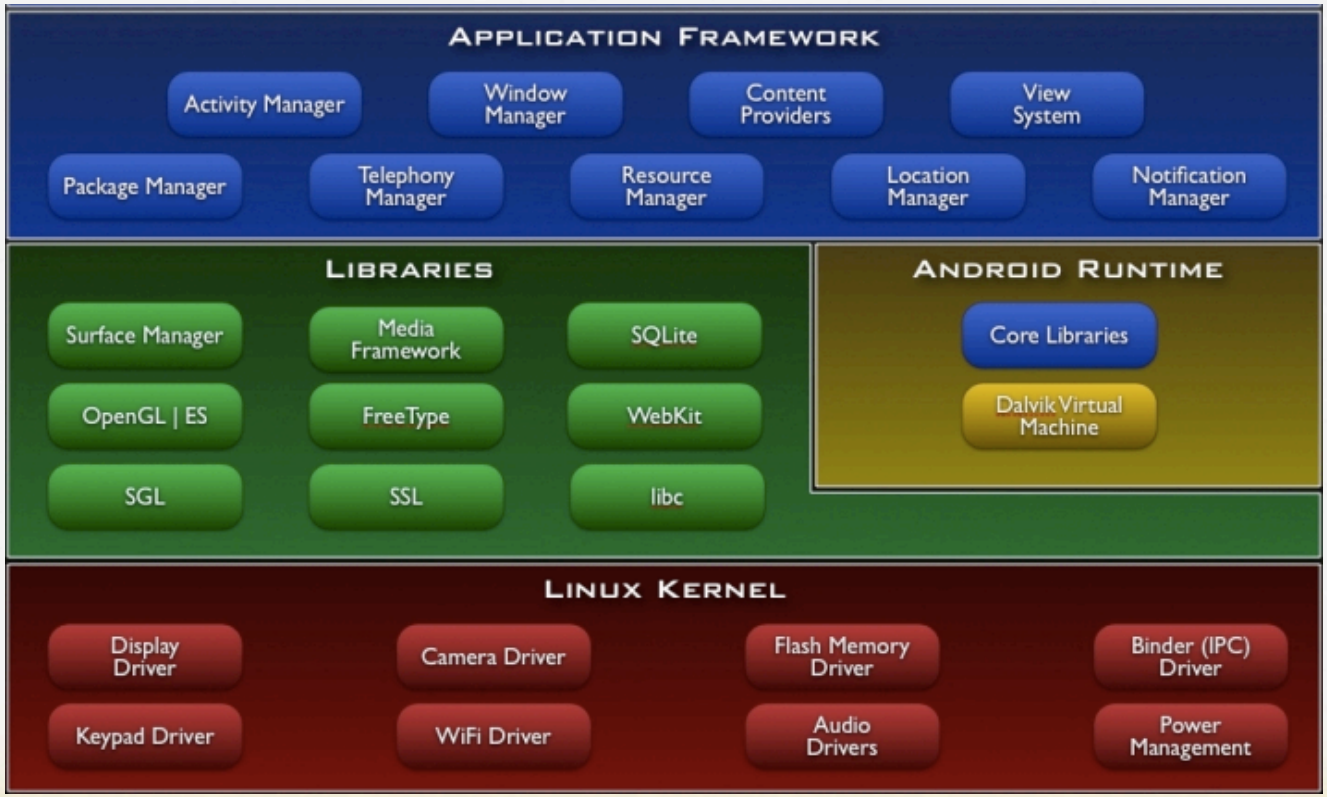
What is the difference between a mobile OS and a desktop/server OS?

ARCHITECTURE

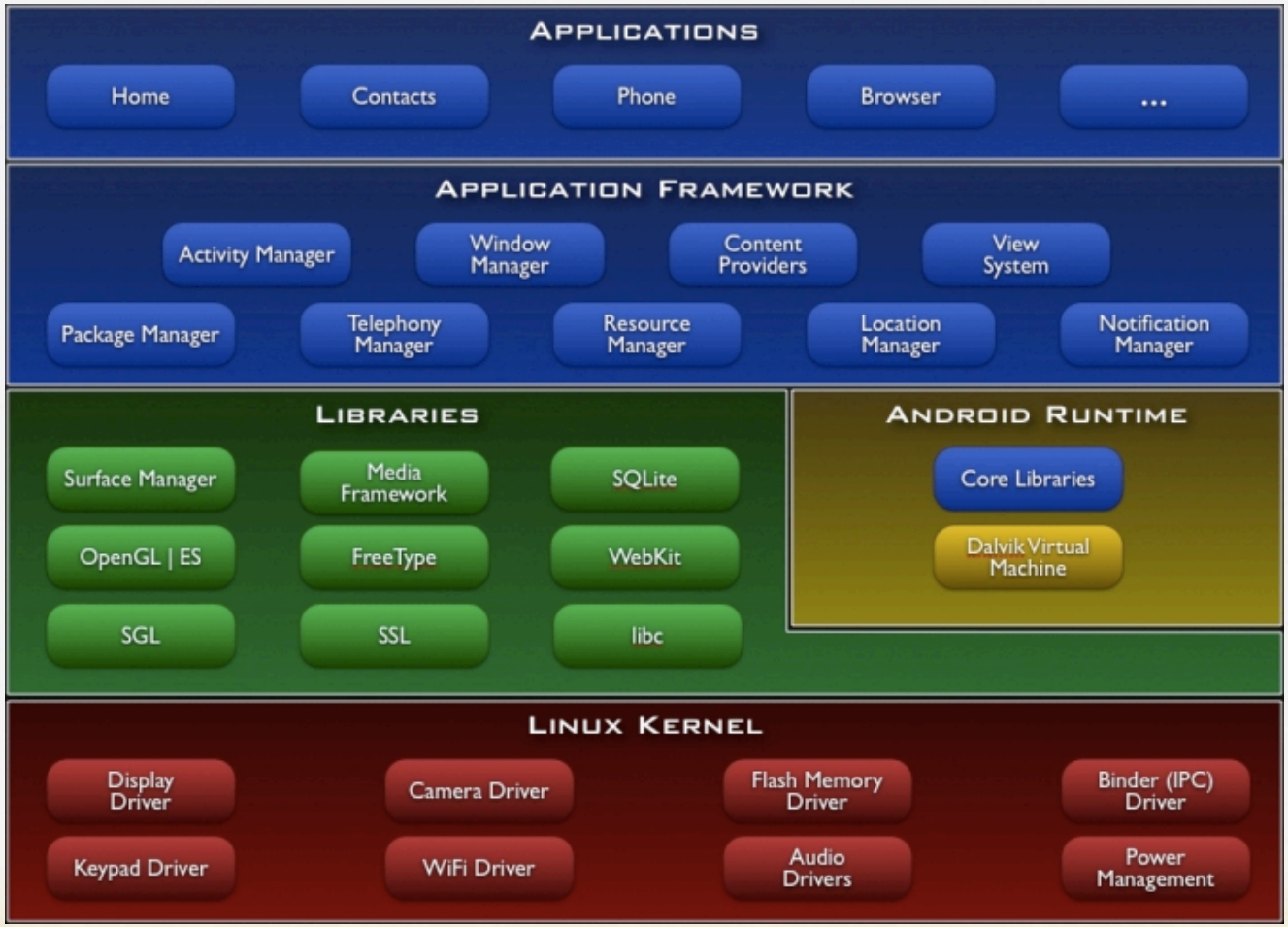
ANDROID


Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

ANDROID


Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

ANDROID


Applications
Home
Contacts
Phone
Browser
...
Application Framework
Activity Manager
Window Manager
Content Providers
View System
Package Manager
Telephony Manager
Resource Manager
Location Manager
Notification Manager
Libraries
Surface Manager
Media Framework
SQLite
OpenGL | ES
FreeType
WebKit
SGL
SSL
libc
Android Runtime
Core Libraries
Dalvik Virtual Machine
Linux Kernel
Display Driver
Camera Driver
Flash Memory Driver
Binder (IPC) Driver
Keypad Driver
WiFi Driver
Audio Drivers
Power Management

BASED ON LINUX

- Android uses Linux 2.6 kernel as the hardware abstraction
- What are the essences an OS should provide?
  - Memory management, process management, IPC
    - No virtual memory; specially implemented IPC
- Drivers and architecture support
  - How to port Android to a new device?
- Using Linux vs. Writing a new OS from scratch
  - Do all Linux kernel implementations work well on mobile devices?

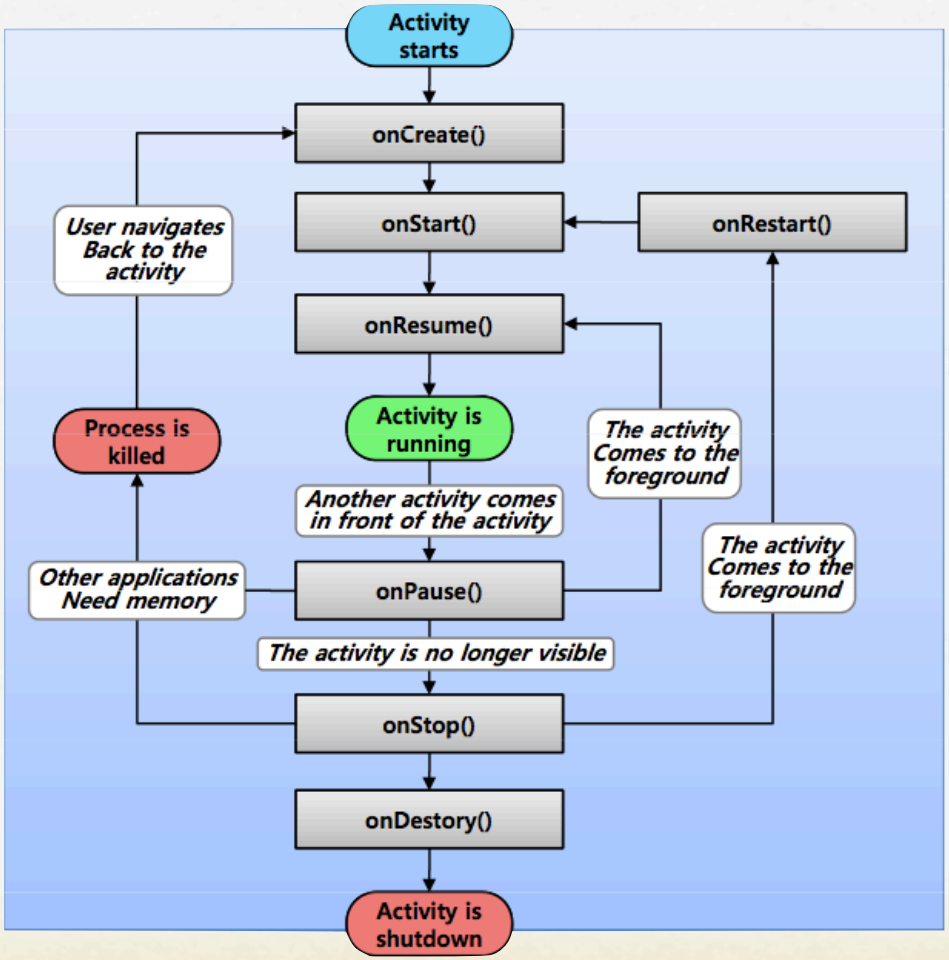
APPLICATION LIBRARY

- GNU libs (glibc) is too big and complicated for mobile phones, so Android implements its own special version of libc - *Bionic libc*:
  - Smaller size - 200K (glibc is more than 400K)
  - Strip out some complicated C++ features, the most significant one - no C++ exception!
  - Very special and small pthread implementation, heavily based on kernel futexes
  - Bionic libc does *not* fully support POSIX and is *not* compatible with glibc
    - which means ...?

PROCESS MANAGEMENT

- What's the difference between mobile apps cycle and desktop apps cycle?
- Two key principles
  - Android usually do not kill an app, i.e. apps keep running even after you switch to other apps
  - Android kills apps when the memory usage goes too high, but it saves app state for quick restart later on
- Do they make sense to mobile apps?

APPLICATION LIFE CYCLE



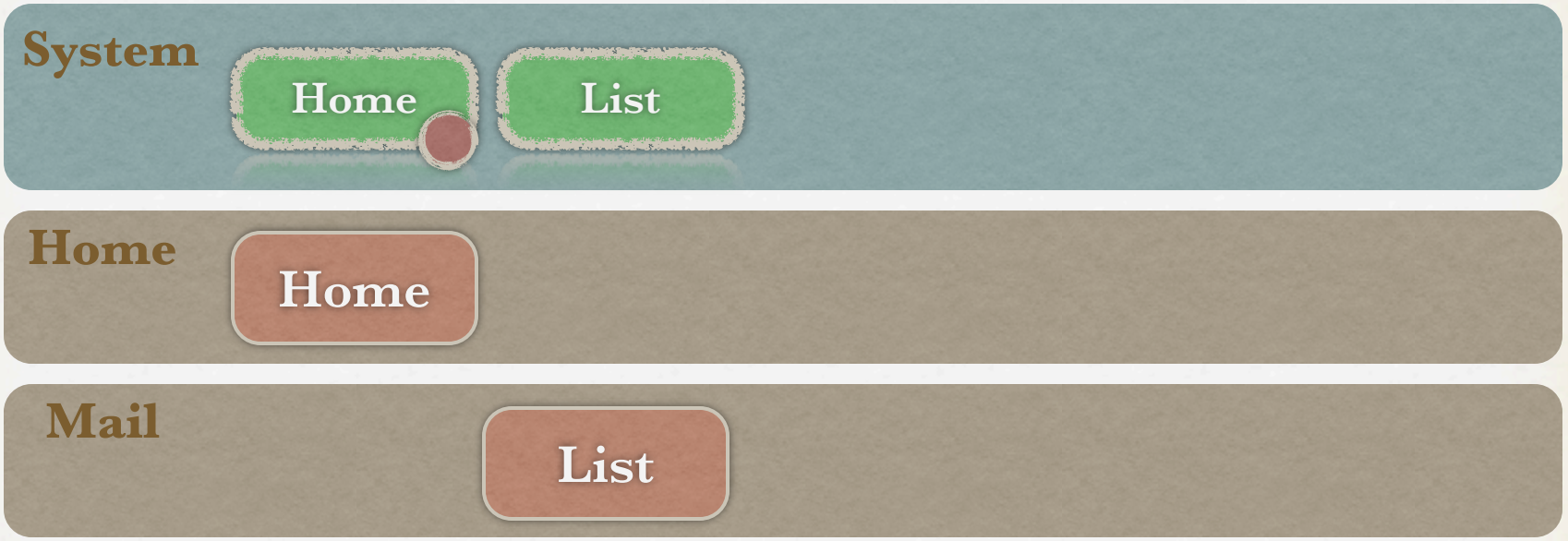
EXAMPLE

System Home

Home Home

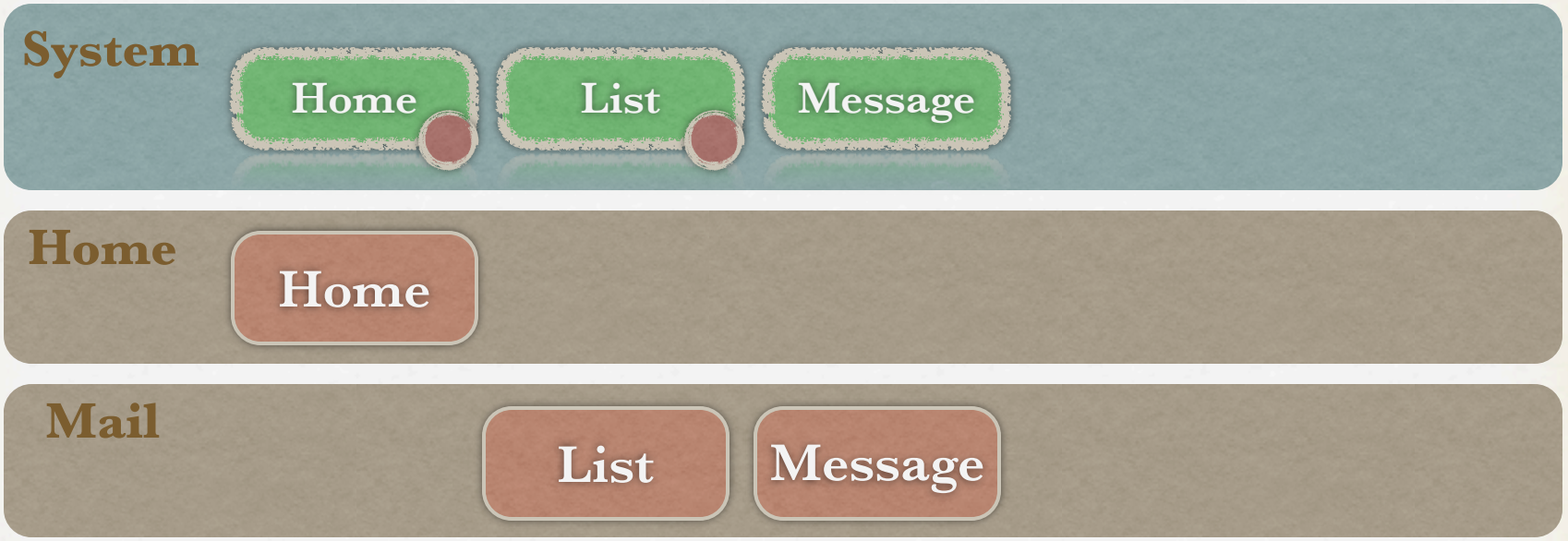
At the “Home” screen

EXAMPLE



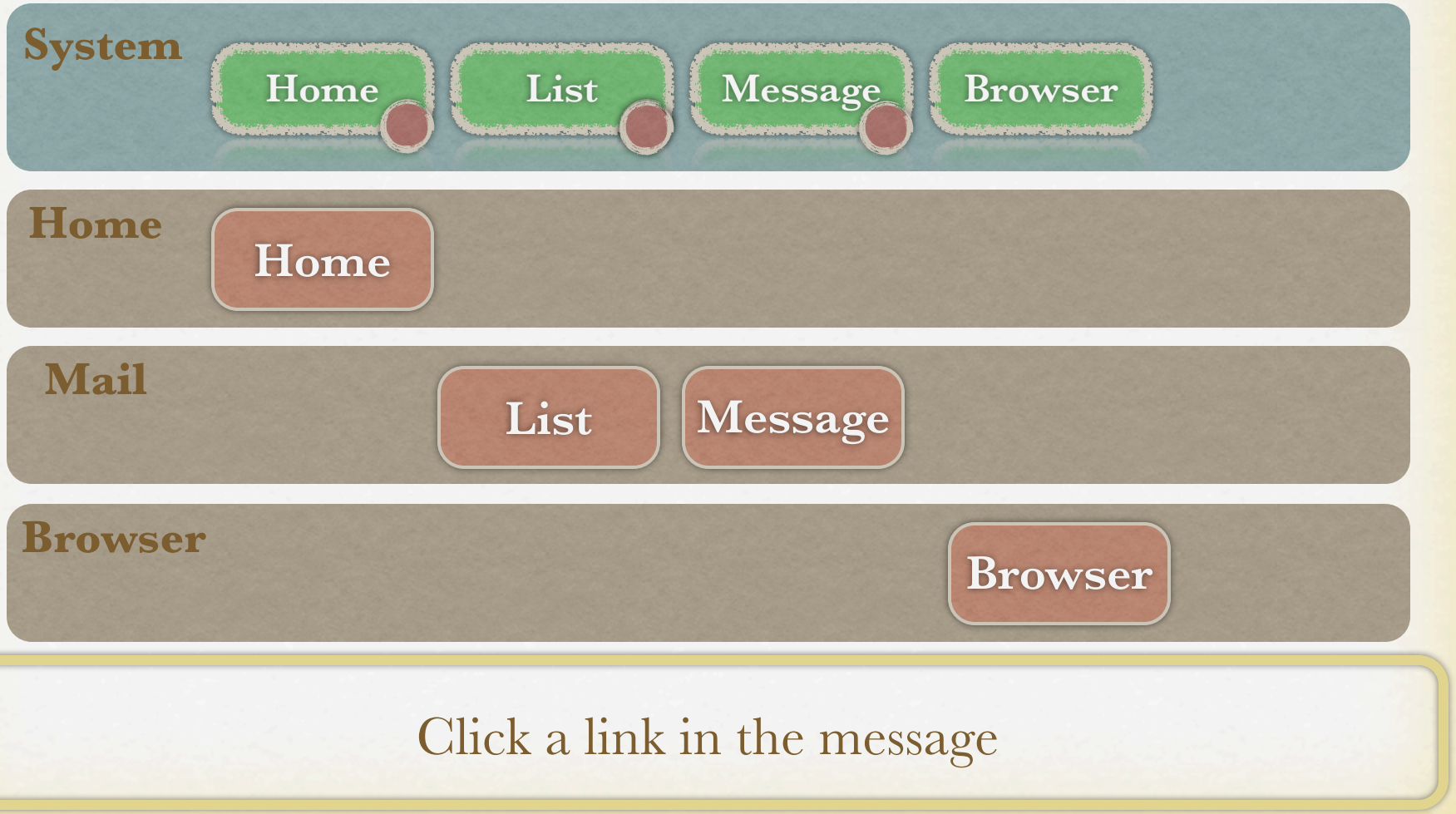
Start the “Mail” app and read the list

EXAMPLE



Click on one of the message and see its content

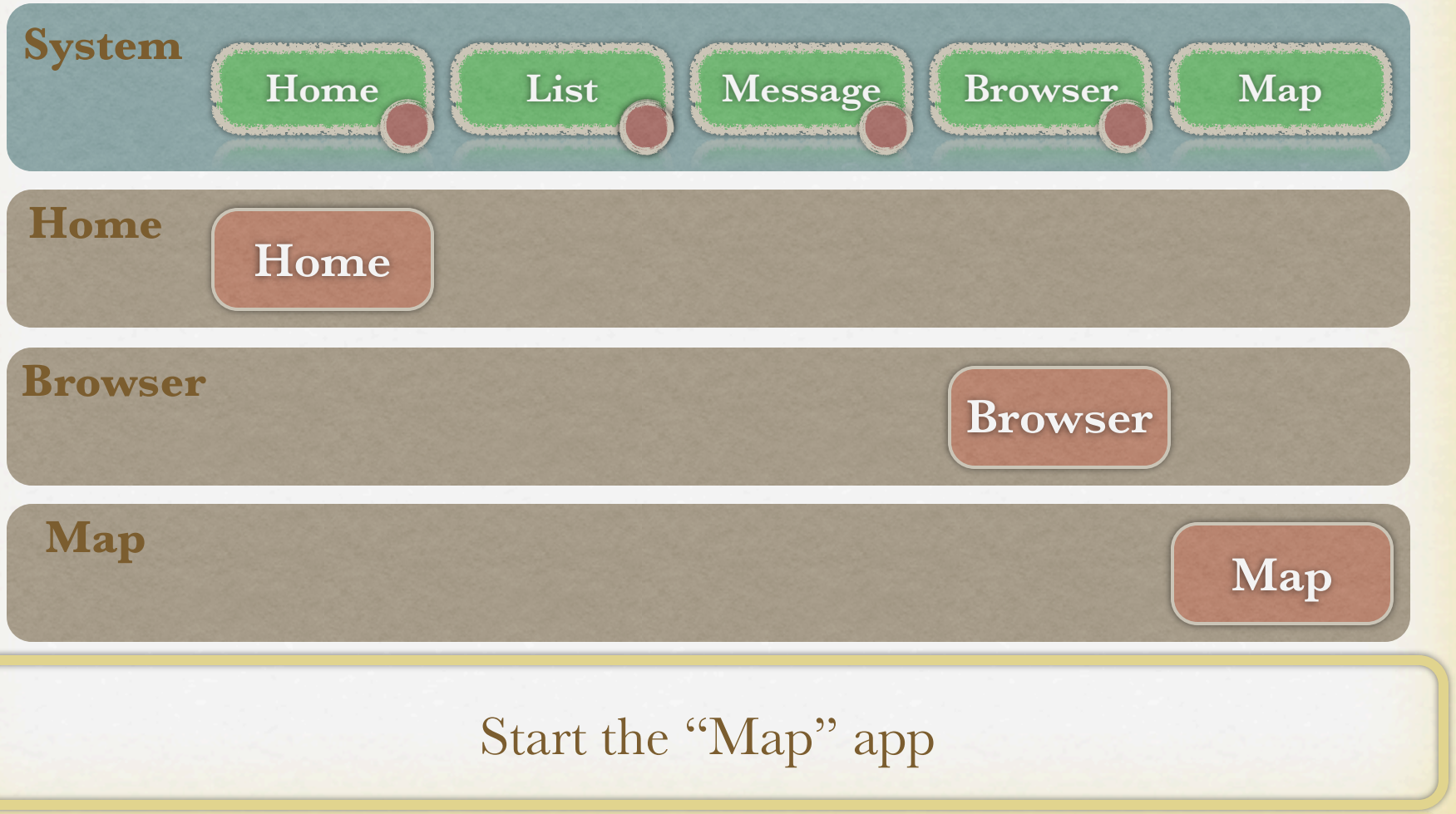
EXAMPLE


System
Home
List
Message
Browser
Home
Home
Mail
List
Message
Browser
Browser
Click a link in the message

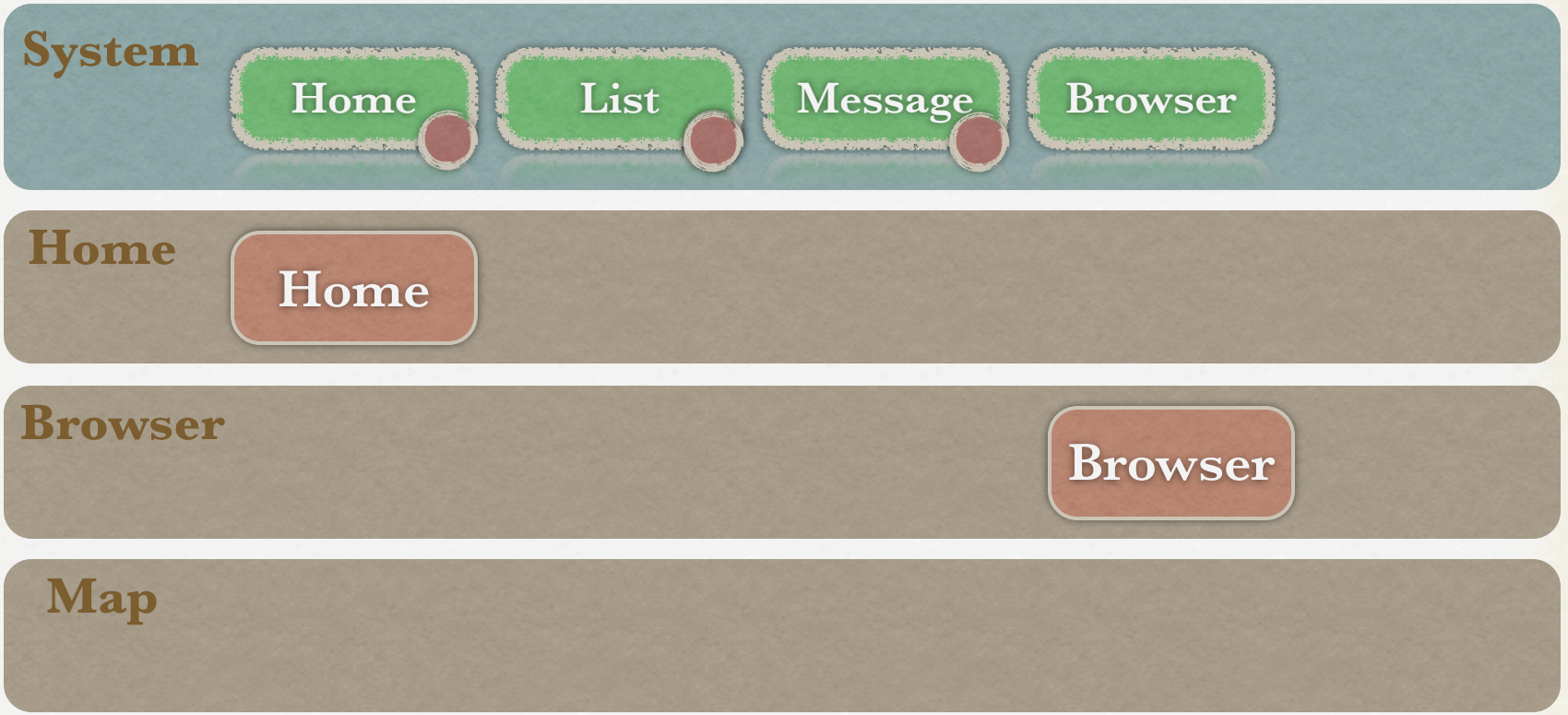
EXAMPLE

Now we have enough space to start the “Map” app

EXAMPLE

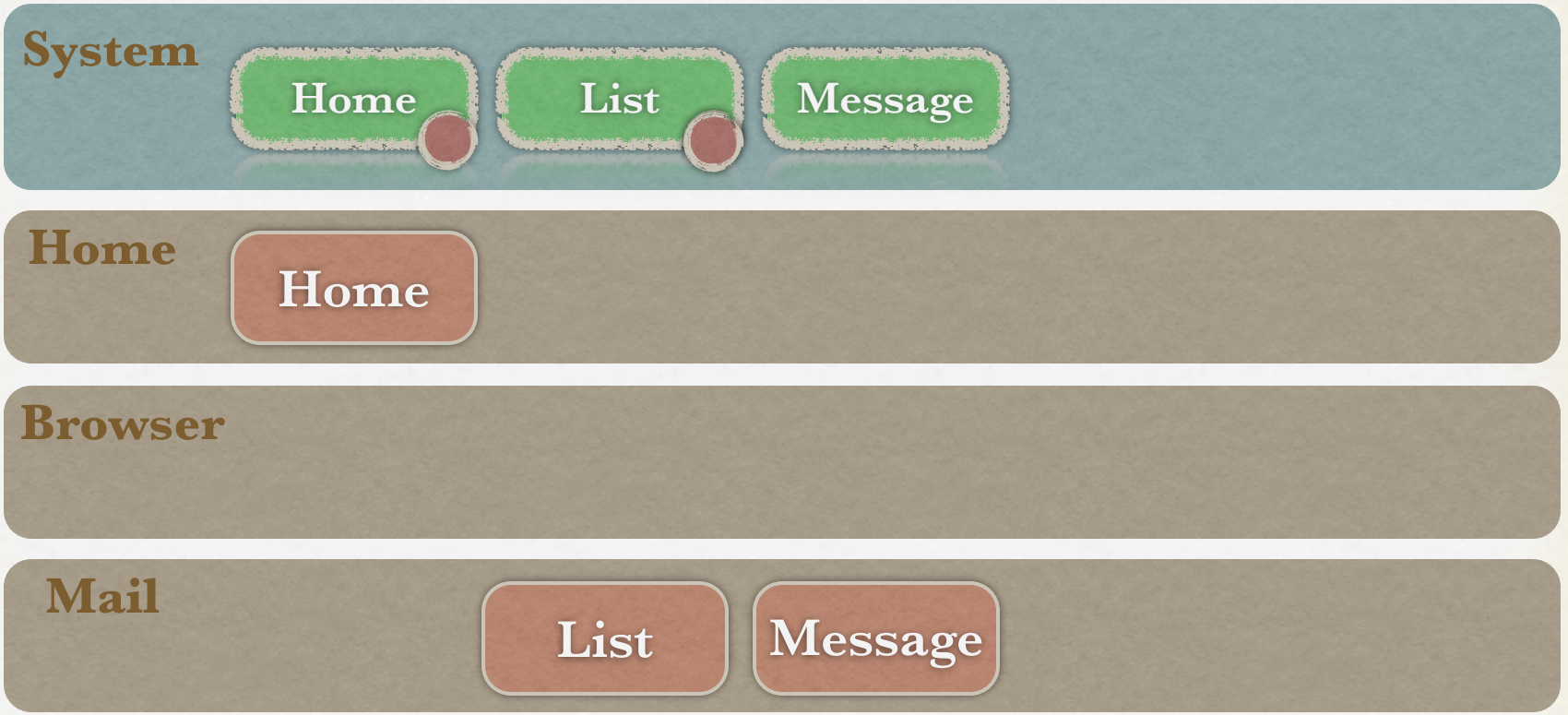

System
Home
List
Message
Browser
Map
Home
Home
Browser
Browser
Map
Map
Start the “Map” app

EXAMPLE



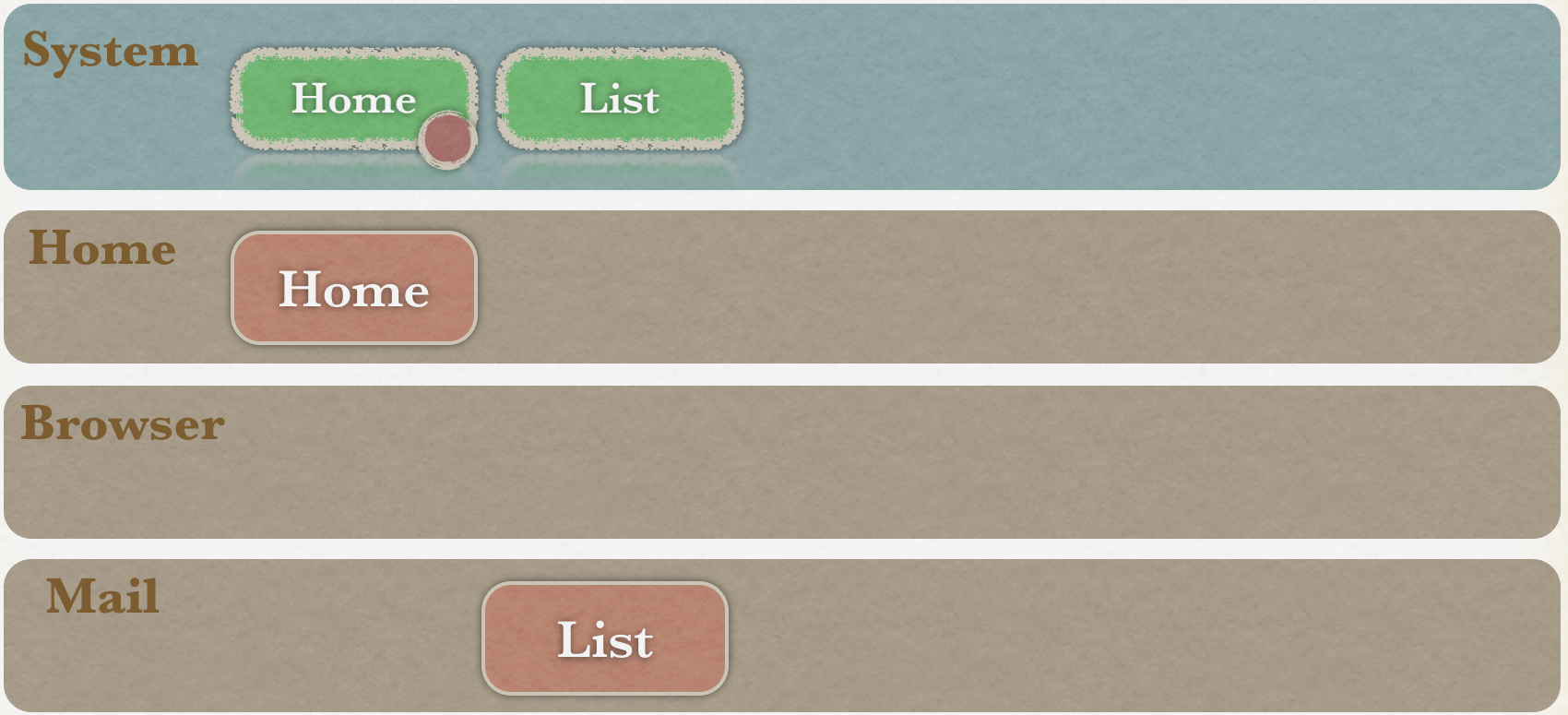
Go back to the browser

EXAMPLE



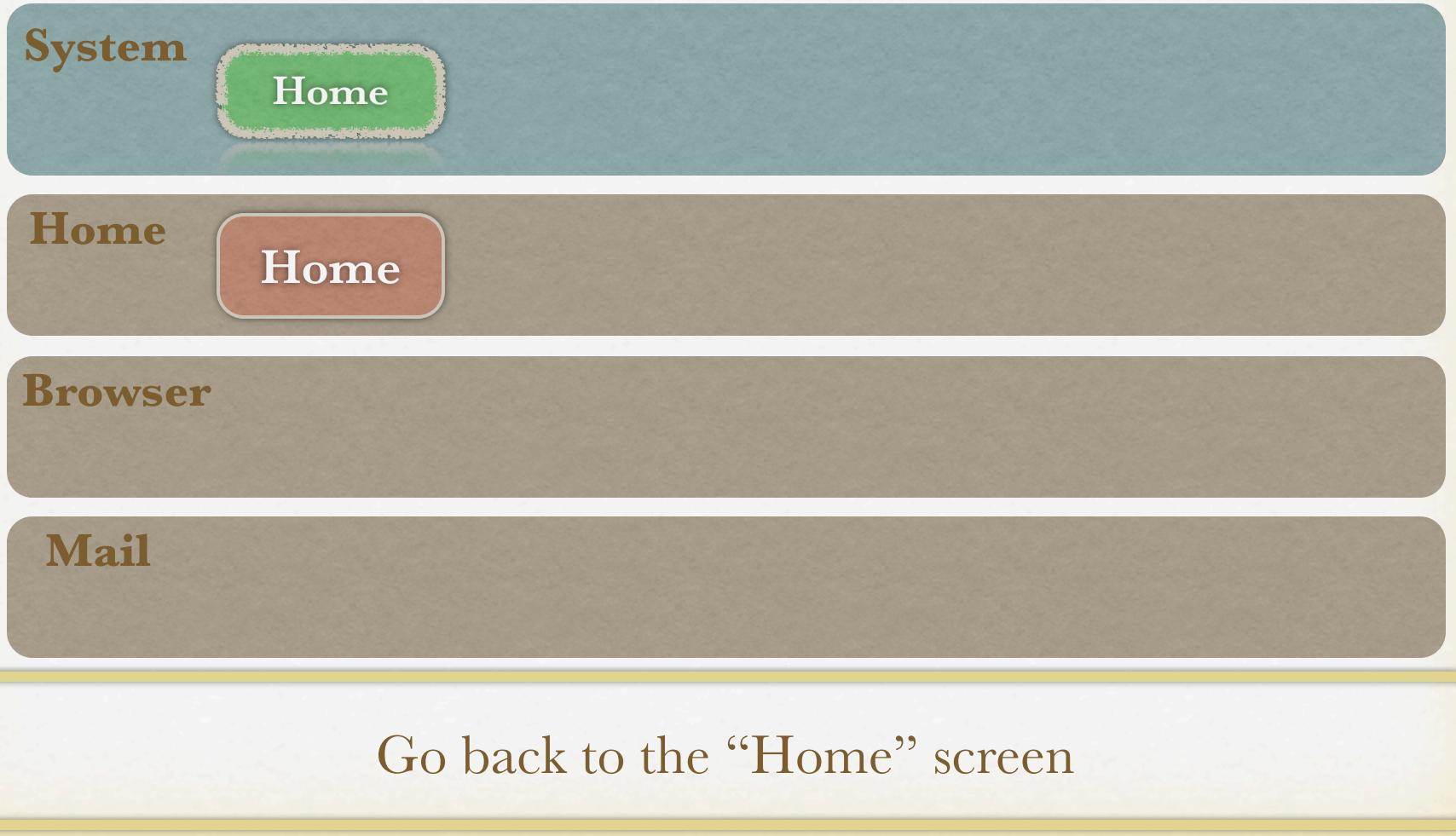
The "Mail" app is *resumed* and shows the previous message

EXAMPLE



Go back to the mail list

EXAMPLE



Go back to the “Home” screen

DEBATE

Swapping model
VS.
Android's life-cycle model

DISK I/O

	Flash	Hard Disk Drive
Random access	~0.1ms	5-10ms
File fragment impact	No	Greatly impacted
Total power	1/2 to 1/3 of HDD	up to 15+ watts
Reliability	Reliable	Less reliable due to mechanical parts
Write longevity	**Limited number of writes**	Less of a problem
Capacity	<= 512GB	2-3TB
Price	$1.5-2 / GB	$0.1-0.2 / GB

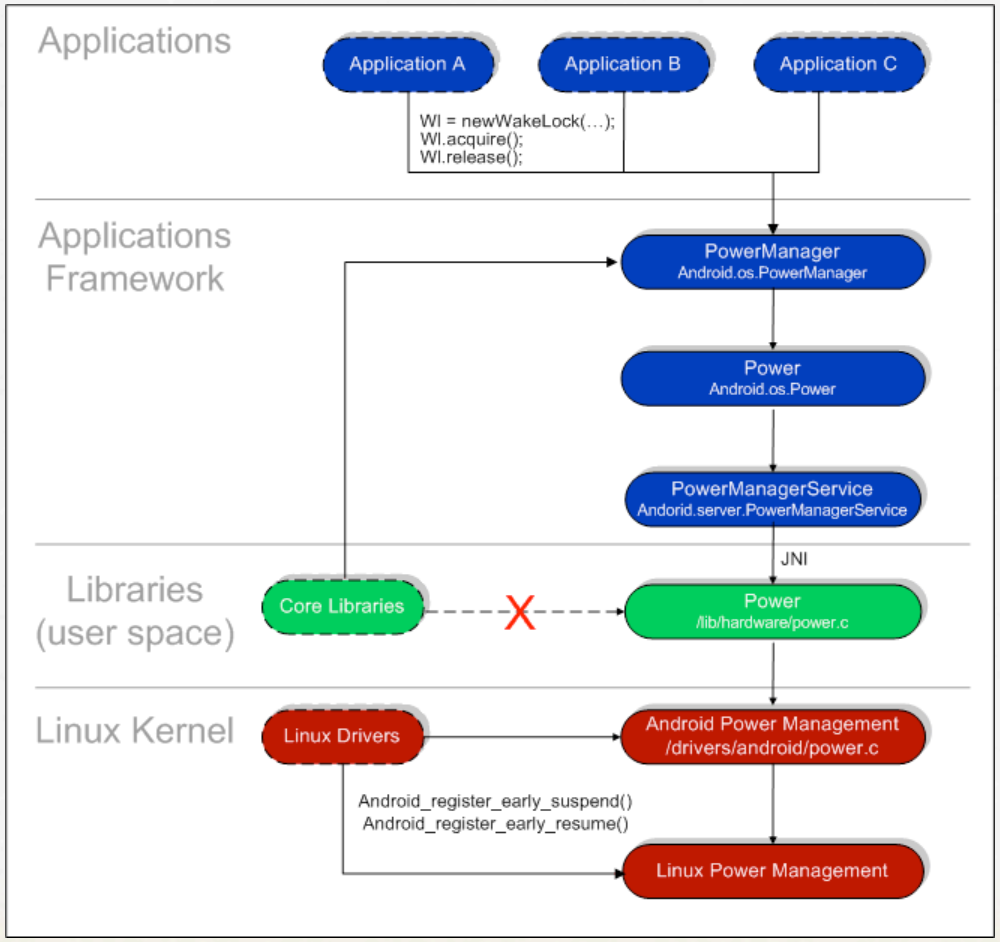
LIMITED WRITES?

- Flash drives have the well-known problem of limited number of writes in the life time - 10,000~100,000 times. Solution?
  - What can applications do?
  - How about operating system?
  - Controllers?
  - Hardware?

MEMORY MANAGEMENT

- Linux kernel does most of the job
  - Page-based memory management
  - Virtual address to physical address mapping
  - **NO** virtual memory
    - Why do we still need “virtual to physical” address mapping?
    - Why does Android not support virtual memory?

POWER MANAGEMENT



DALVIK VM

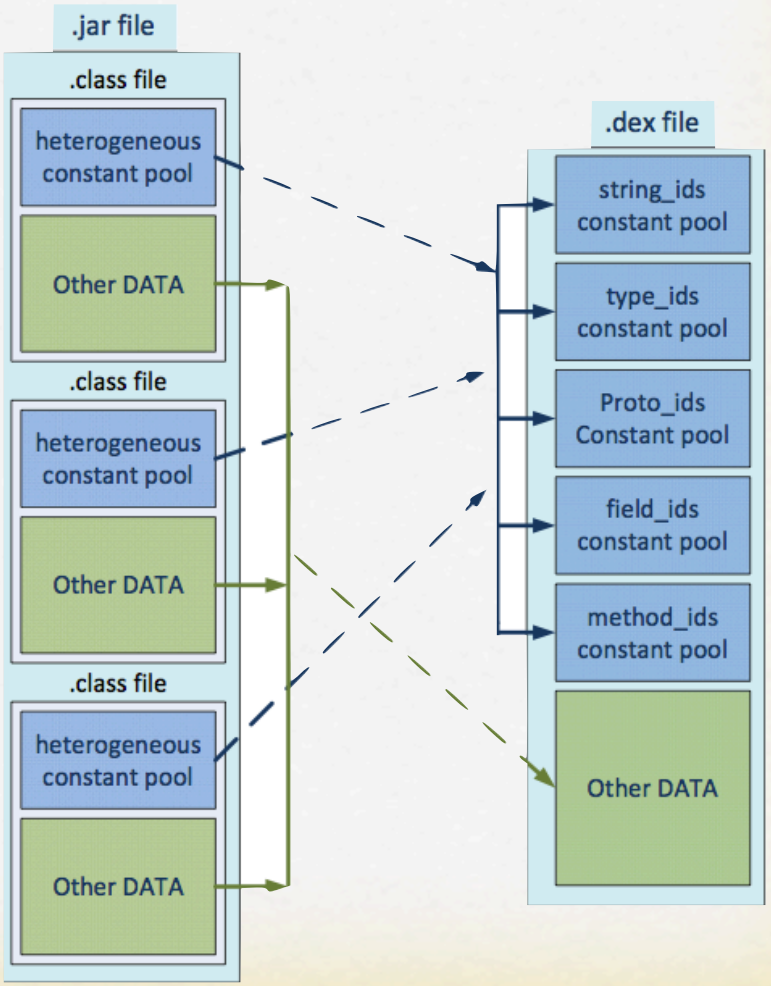
Why does Android let developers use Java?

DALVIK VM

- A special Java virtual machine (VM) designed to run with limited system resource
- Memory efficiency
- Register machine vs. Stack machine (modern JVM)
  - fewer instructions, faster execution
  - why does the number of instructions matter?
- Running multiple VMs more efficiently

DEX FILE

- Java class files are converted into ".dex" files that Dalvik executes
- Java byte-code is converted into Dalvik byte-code during this process

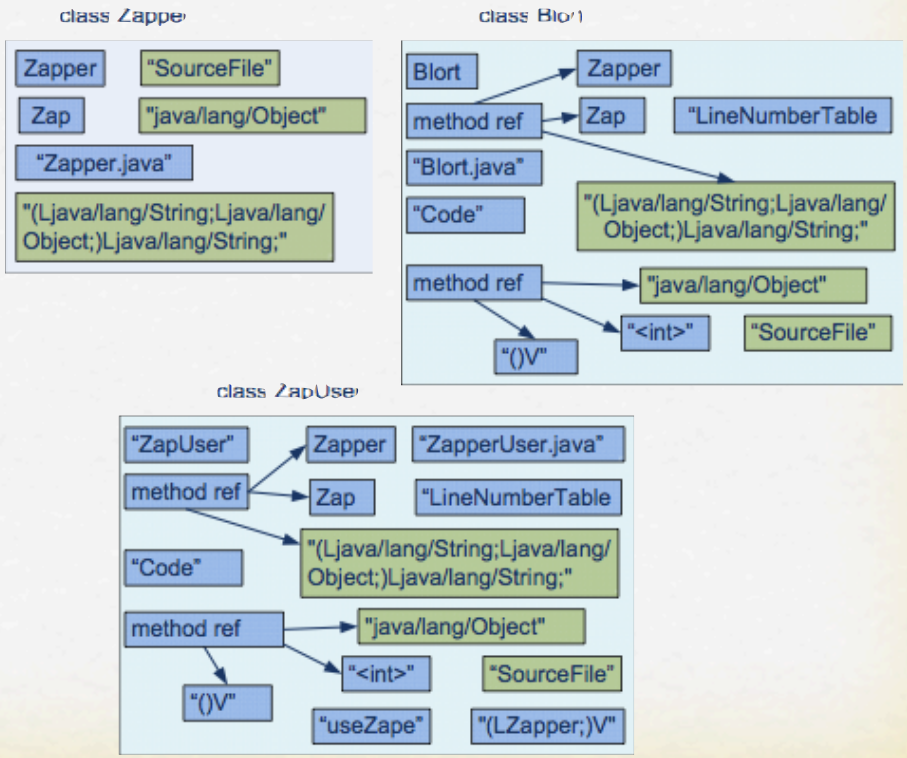


MEMORY EFFICIENCY

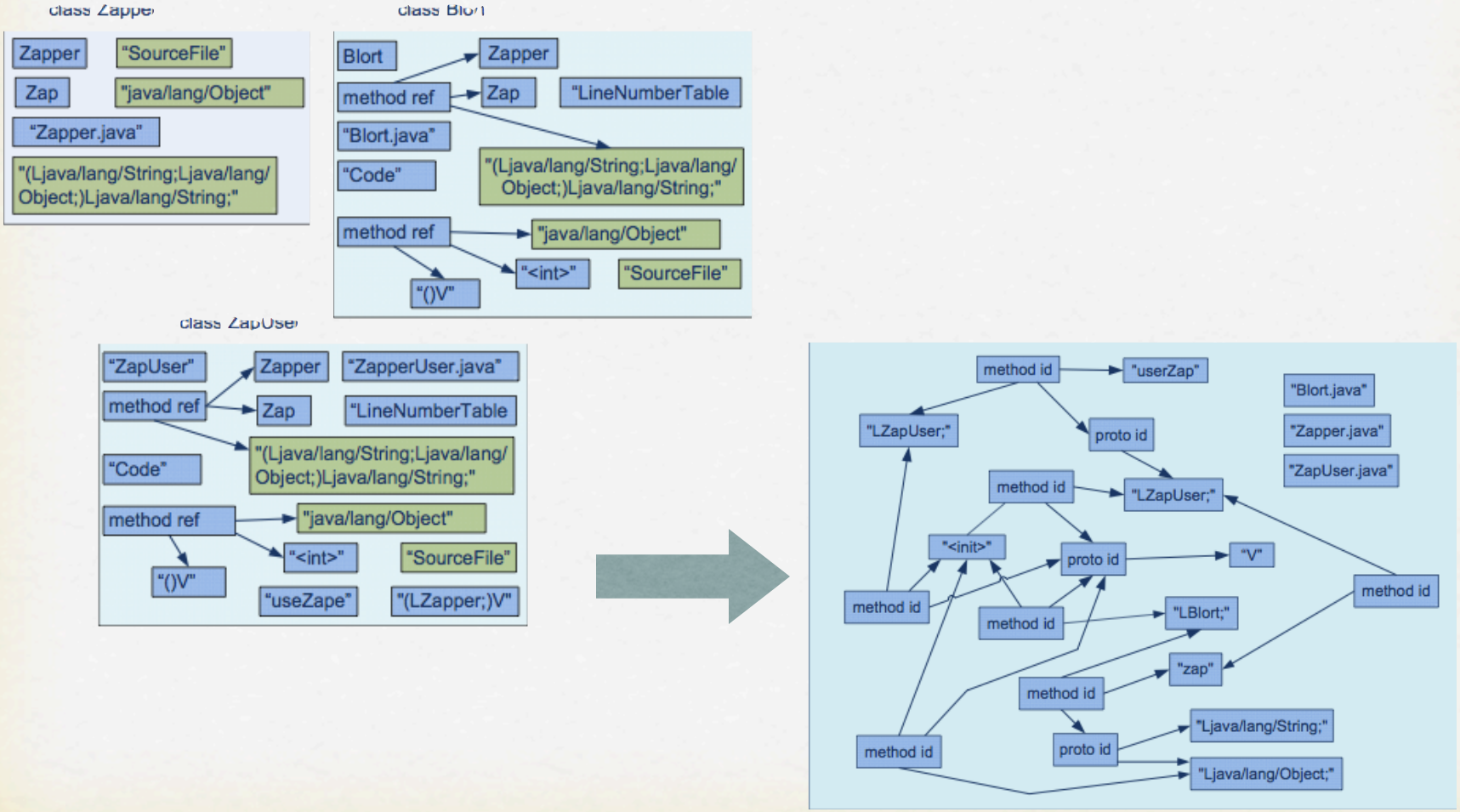
- Shared constant string pool
- Share clean (even some dirty) memory between processes as much as possible
- “.dex” files are mapped as read-only by mmap()
- Memory efficient JIT implementation
  - JIT itself is about 100K
  - Code cache and supporting data structure takes another 100K for each application

SHARED STRING POOL

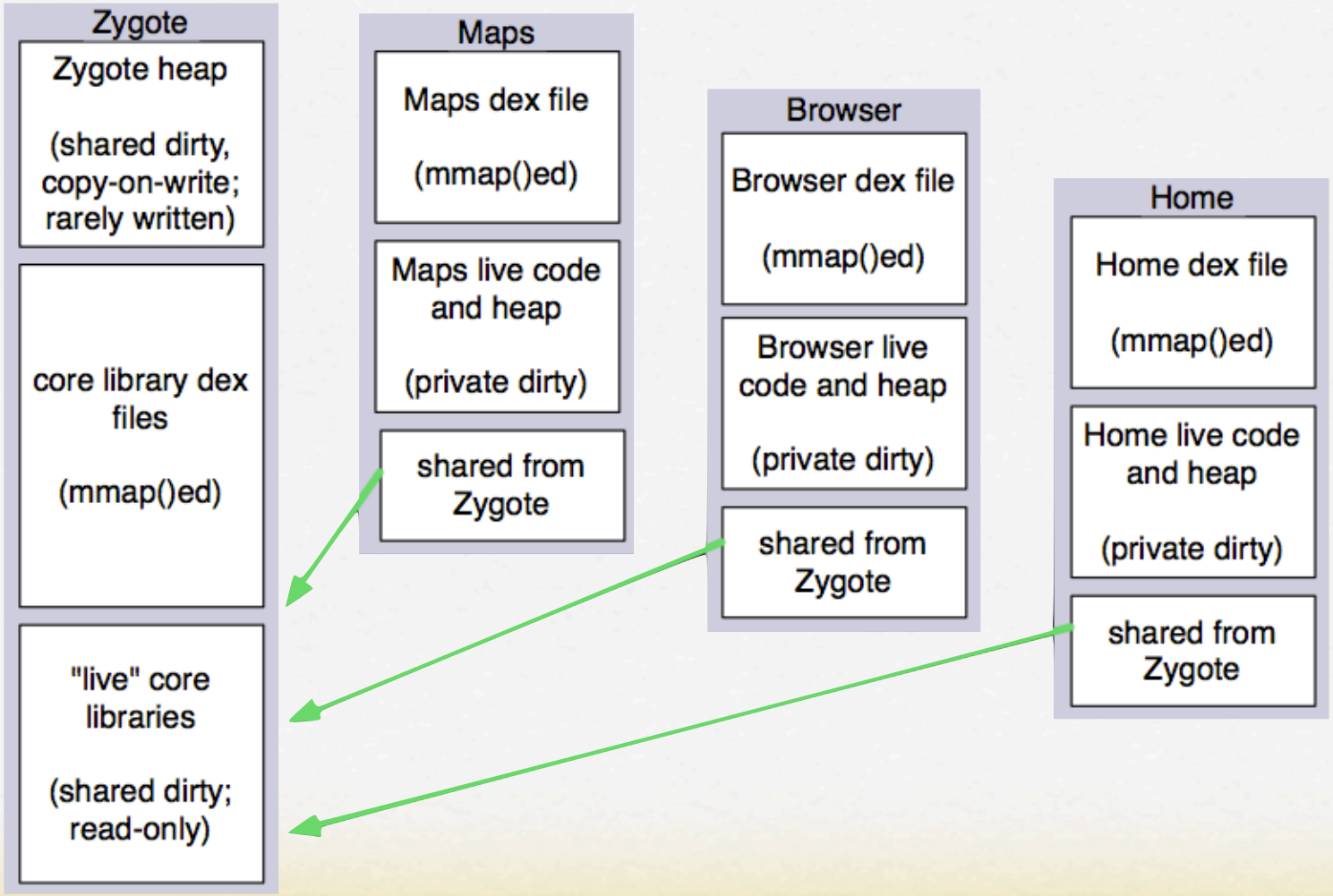
```
public interface Zapper {
    public String zap(String s, Object o);
}

public class Blort implements Zapper {
    public String zap(String s, Object o) {
        ....
    }
}

public class ZapUser {
    public void useZap(Zapper z) {
        z.zap(...);
    }
}
```



SHARED STRING POOL



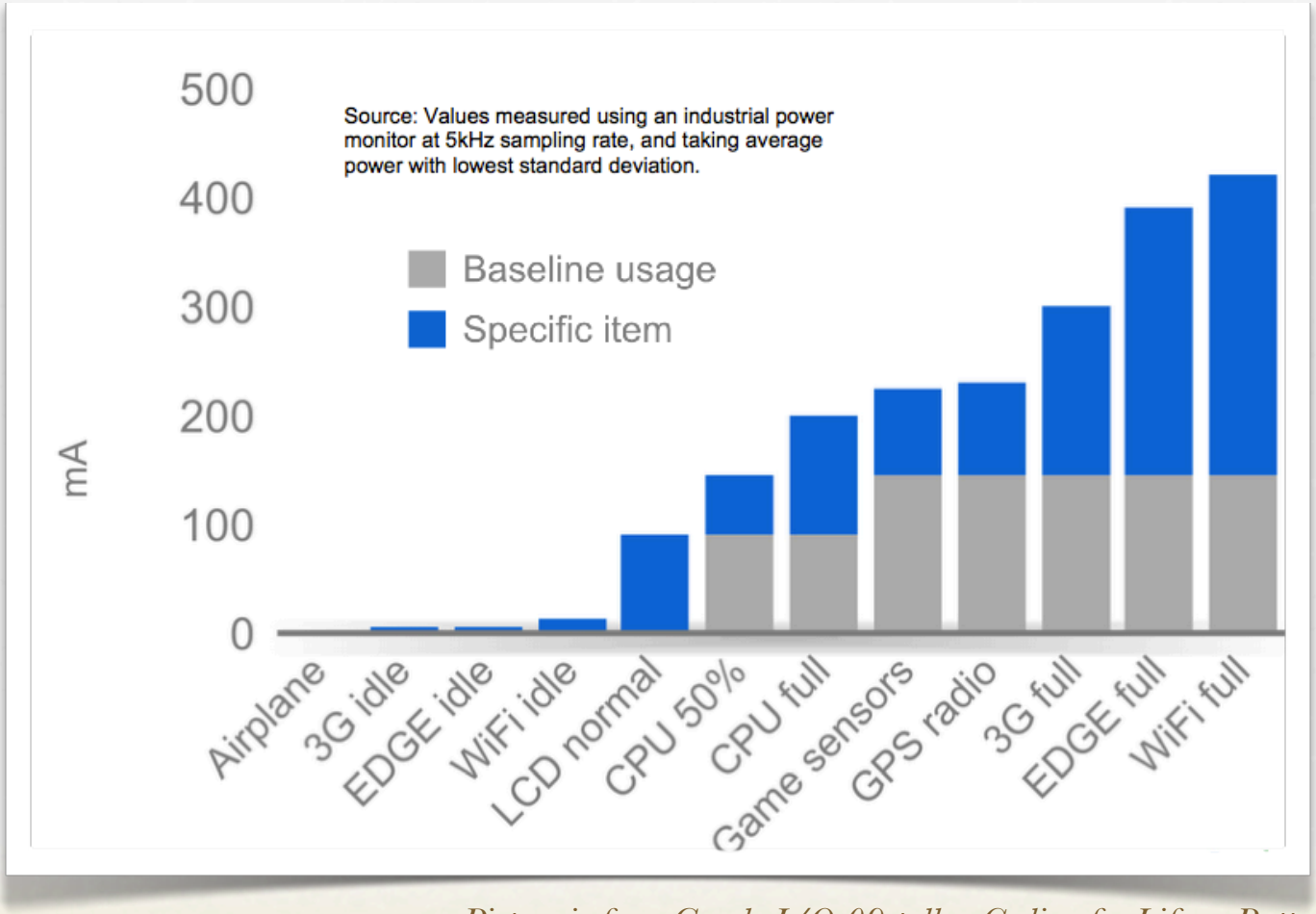
SHARED MEMORY



PROGRAMMING MODEL

- Each application is running in its own process
- An application can have one or more components:
  - activities, services, broadcast receivers and content providers
- A task (*an "application" from user's point of view*) consists of several activities from one or multiple applications
- An application keeps running until the system kills it because of memory shortage

POWER SAVING



Picture is from Google I/O 09 talk - Coding for Life -- Battery Life, That Is

GZIP TEXT DATA

- Use GZIP for text data whenever possible
  - Compressing is implemented by native code

```
import java.util.zip.GZIPInputStream;

HttpGet request =
    new HttpGet("http://example.com/gzipcontent");
HttpResponse resp =
    new DefaultHttpClient().execute(request);
HttpEntity entity = response.getEntity();
InputStream compressed = entity.getContent();
InputStream rawData = new GZIPInputStream(compressed);
```

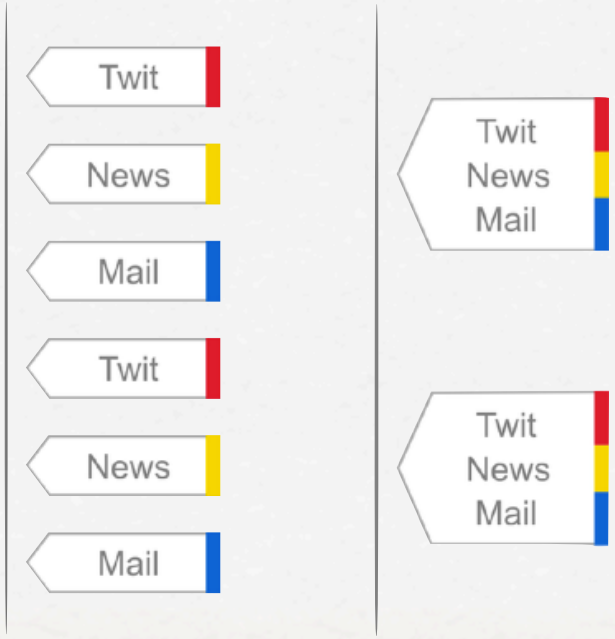
CHECK NETWORK TYPE

- Wifi and 3G are much more energy efficient, so wait for Wifi or 3G when transferring big chunk of data

```
// Only update if WiFi or 3G is connected and not roaming
int netType = info.getType();
int netSubtype = info.getSubtype();

if (netType == ConnectivityManager.TYPE_WIFI) {
    return info.isConnected();
} else if (netType == ConnectivityManager.TYPE_MOBILE
        && netSubtype == TelephonyManager.NETWORK_TYPE_UMTS
        && !mTelephony.isNetworkRoaming()) {
    return info.isConnected();
} else {
    return false;
}
```

UPDATE BIN

- Use `setInexactRepeating()` so the system can bin your update together with others



Picture is from Google I/O 09 talk - Coding for Life -- Battery Life, That Is

WORK OFFLOADING

- Naive offloading
  - Speech-to-text, OCR
- More sophisticated offloading - fine-grained offloading
  - MAUI: Making Smartphones Last Longer with Code Offload (MobiSys '10)
  - Running two versions of the app on the mobile device and a powerful server
  - Decide when/what to offload on the fly

EFFICIENT CODE

- ```
  for (int i = initializer; i >= 0; i--)
  ```
- ```
  int limit = calculate limit;
  for (int i = 0; i < limit; i++)
  ```
- ```
  Type[] array = get array;
  for (Type obj : array)
  ```
- ```
  for (int i = 0; i < array.length; i++)
  ```
- ```
  for (int i = 0; i < this.var; i++)
  ```
- ```
  Iterable<Type> list = get list;
  for (Type obj : list)
  ```

EFFICIENT CODE

- Try to rest for the most of the time
  - be nice to other processes
- Avoid allocation
  - short-lived objects need to be garbaged collected
  - long-lived objects take precious memory
- Make a method `static` if it does not access member variables
- Avoid internal getter/setters
- Use floating point numbers only when you have to
- Prefer `int` over `enum`
- Use `static final` for constants