Department of Information Engineering, CUHK
MScIE – 2nd Semester, 2015/16

IEMS 5722
Mobile Network Programming and Distributed Server Architecture

Lecture 10
Cloud Computing & Cloud Services

Lecturer: Albert C. M. Au Yeung

17th March, 2016

Brief Introduction to Cloud Computing

Cloud Computing

What is cloud computing?

Servers

What people do when they need to run a Web application?

The first Web Server (a NeXT computer)

Data Centres

What people do when they need to run a Web application?

Data centres

Data Centre Services

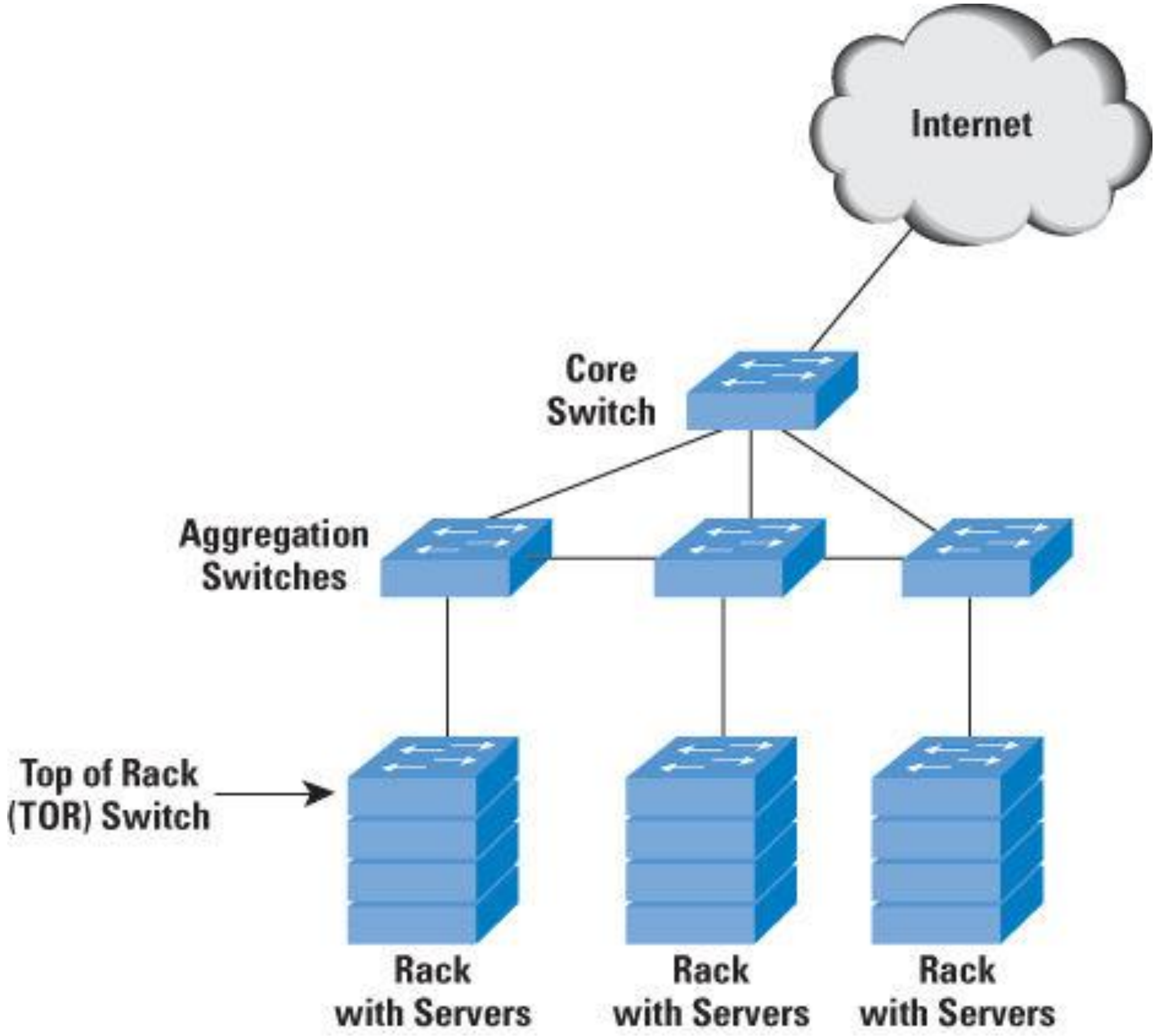


Figure from "Cloud Computing - A Primer - The Internet Protocol Journal", The Internet Protocol Journal, Volume 12, No.3

Data Centre Services

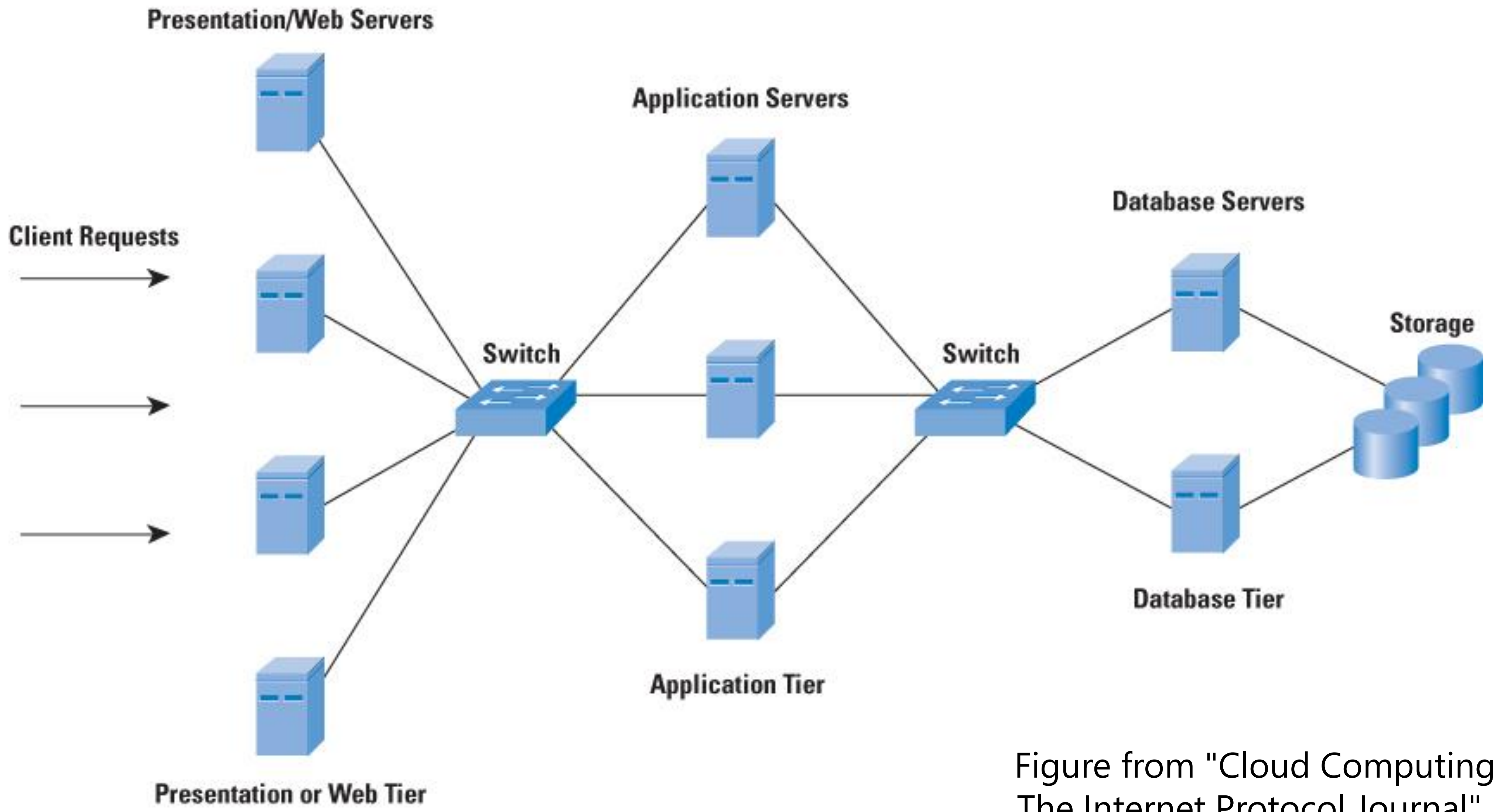
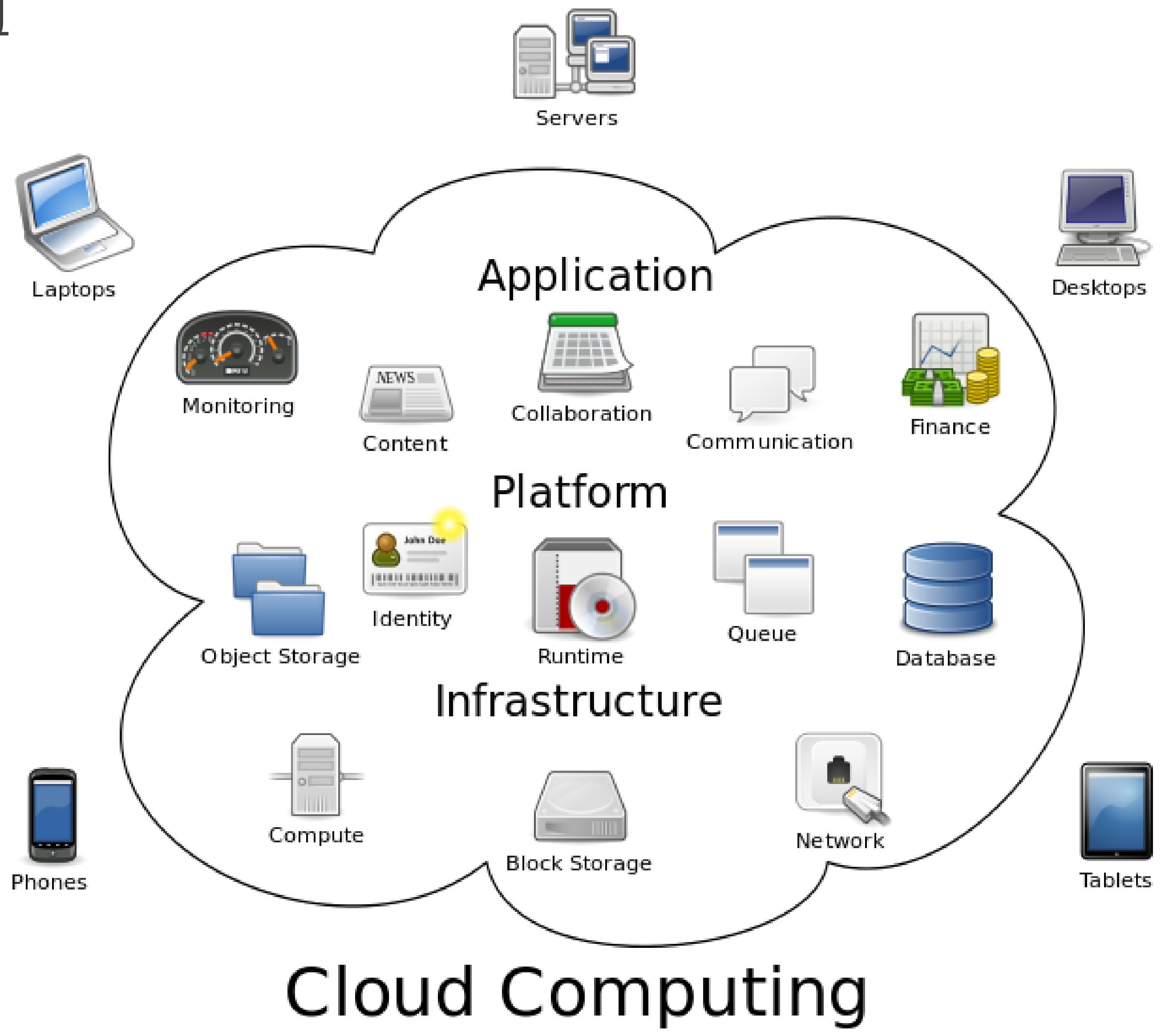


Figure from "Cloud Computing - A Primer - The Internet Protocol Journal", The Internet Protocol Journal, Volume 12, No.3

Cloud Computing



https://en.wikipedia.org/wiki/Cloud_computing

Cloud Computing - Definition

NIST (National Institute of Standards and Technology)

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

Ref: http://www.nist.gov/itl/cloud/

Cloud Computing - Definition

John McCarthy

(Invented the term "Artificial Intelligence")

The first to suggest publicly (in 1961 in a speech given to celebrate MIT's centennial) that computer time-sharing technology might result in a future in which computing power and even specific applications could be sold through the utility business model (like water or electricity).

Cloud Computing - Definition

Deployment Models

Public Cloud

Community Cloud

Hybrid Cloud

Private Cloud

Service Models

Infrastructure as a Service

Platform as a Service

Software as a Service

Characteristics

On-demand Self-service

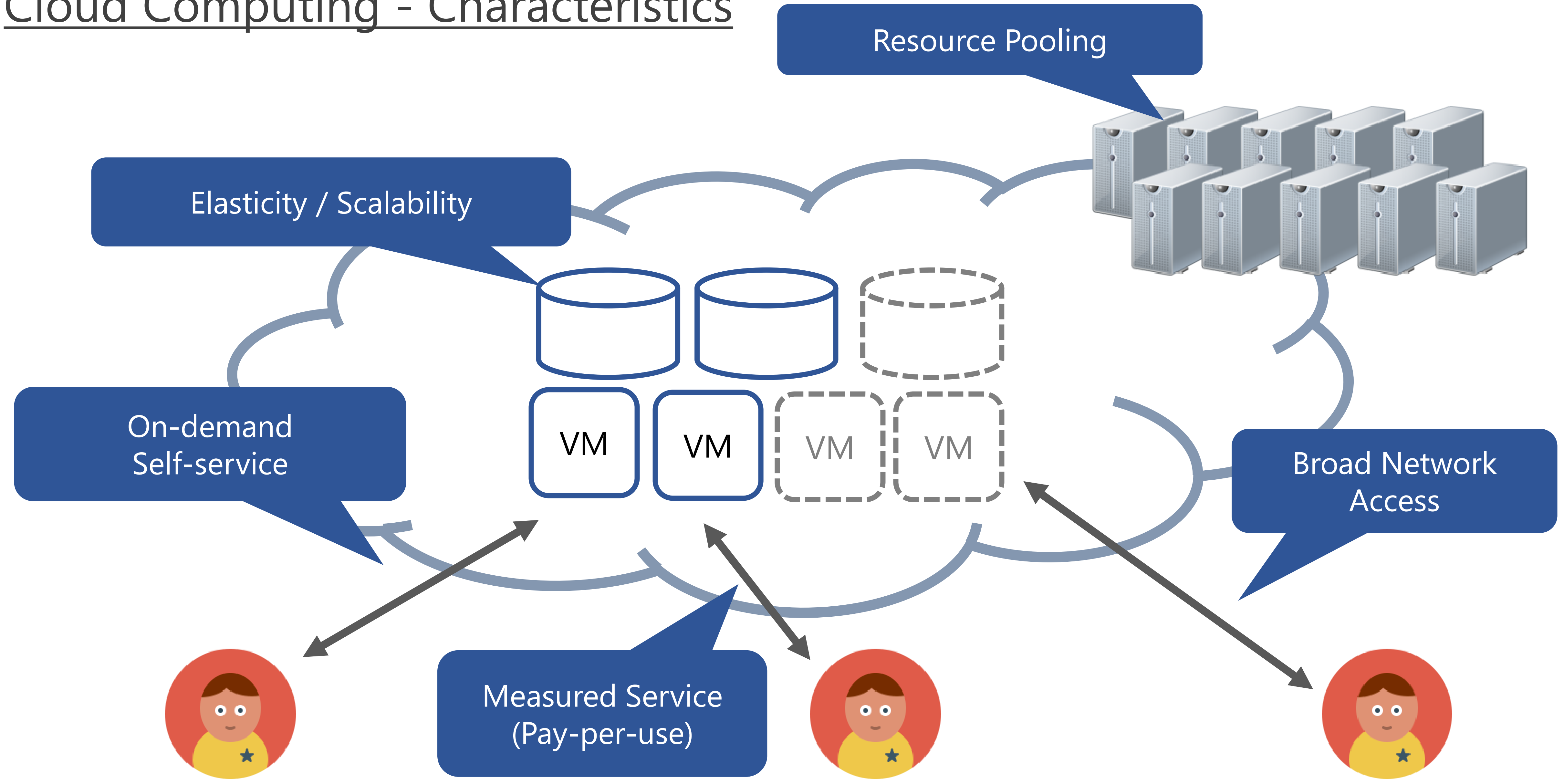
Broad Network Access

Rapid Elasticity

Resource Pooling

Measured Service

Cloud Computing - Characteristics



Infrastructure-as-a-Service (IaaS)

To provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software (e.g. virtual machines)

Consumers do not manage or control the underlying cloud infrastructure but have control over operating systems, storage, and deployed applications; and possibly limited control of select networking components

Platform-as-a-Service (PaaS)

To deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.

Consumers have control over the deployed applications and possibly configuration settings for the application-hosting environment.

heroku salesforce.com

Windows Azure IBM Bluemix™ Google app engine

Software-as-a-Service (SaaS)

To use the provider's applications running on a cloud infrastructure, which are accessible from various client devices through either a thin client interface.

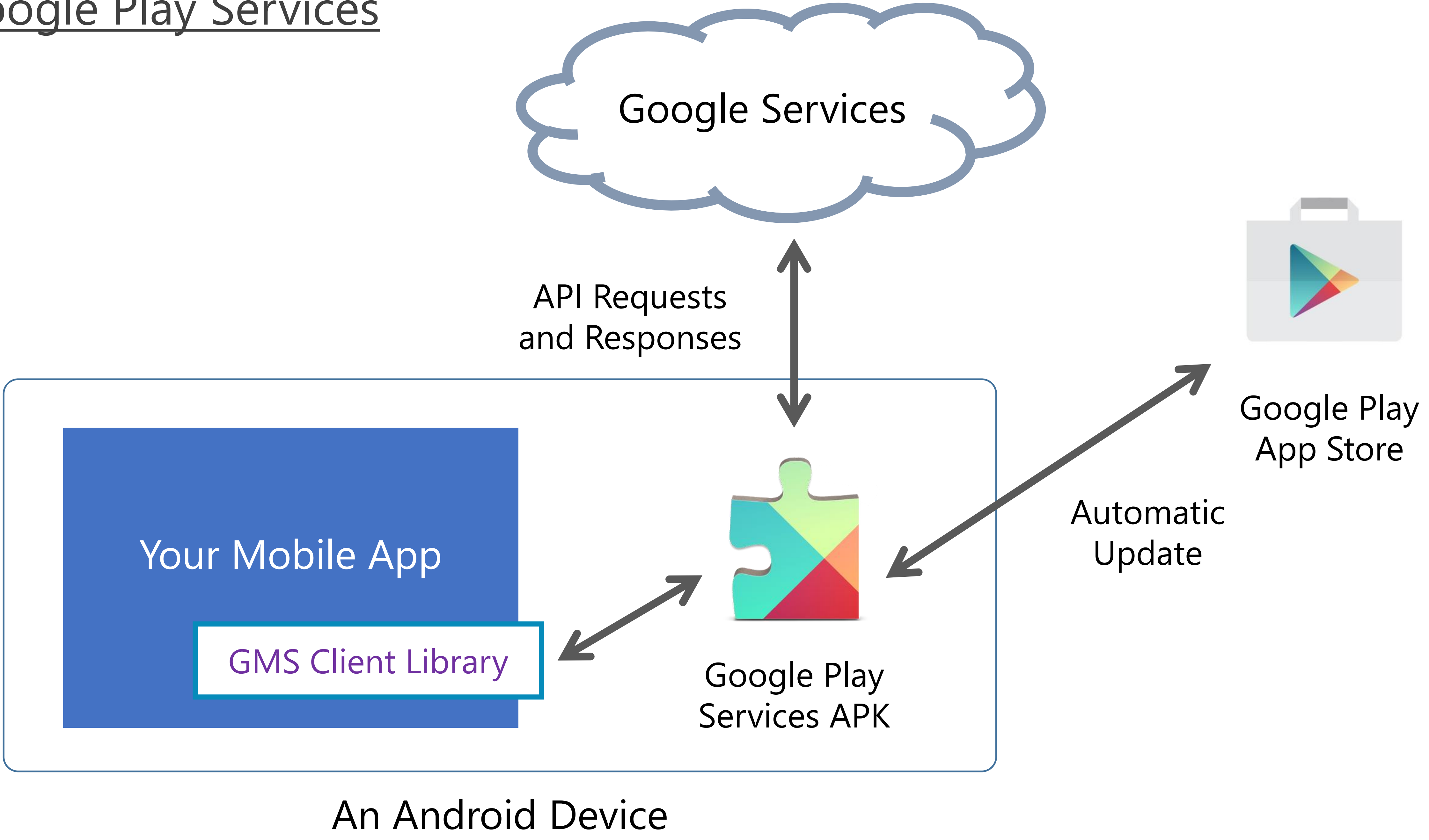
Consumer do not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities

Google Play Services

Google Play Services

- Google Play Services is a set of cloud services and APIs for Android devices
- Version 8.4 released 18 Dec, 2015
- Allows you to integrate various services from Google into the mobile app
- Some of Google's services include
  - Location detection
  - Geocoding and reverse geo-coding
  - Maps API (e.g. navigation, street views)
  - Google drive for cloud storage
  - Google Wallet for online payment

Google Play Services



Google Play Services

Setting up Google Play Services

- Download latest Google Play services SDK
- Update your `build.gradle` file

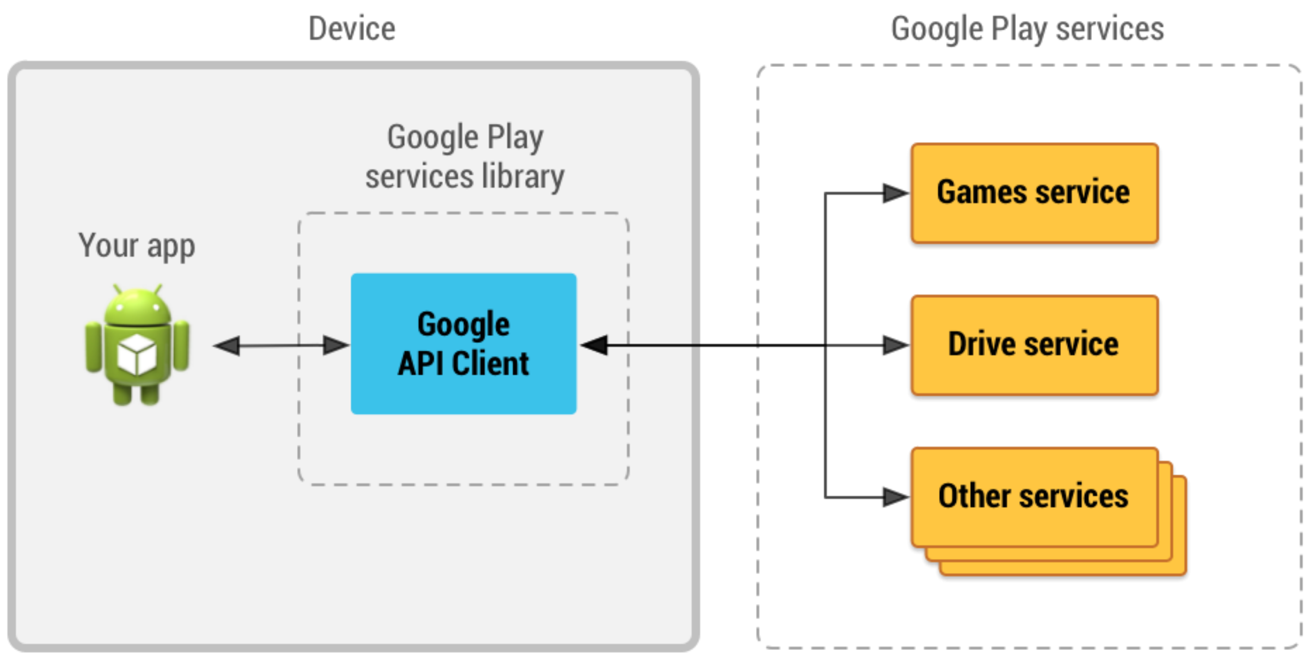
(Refer to the link below)

Reference: https://developers.google.com/android/guides/setup

Accessing Google APIs

To access various Google APIs, you need to create an instance of `GoogleApiClient`

- Acts as a common entry point to all Google services
- Manages network connection between the device and each service



Reference: https://developers.google.com/android/guides/api-client

Accessing Google APIs

You can create an instance of `GoogleApiClient` with connection to specific APIs by using a builder:

```java
public class MyActivity extends FragmentActivity
implements OnConnectionFailedListener {

    private GoogleApiClient mGoogleApiClient;

    @Override protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Create a GoogleApiClient instance
        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .enableAutoManage(this /* FragmentActivity */,
                              this /* OnConnectionFailedListener */)
            .addApi(Drive.API)
            .addScope(Drive.SCOPE_FILE)
            .build();
        // ...
    }

    @Override public void onConnectionFailed(ConnectionResult result) {
        // An unresolvable error has occurred and a connection to Google
        // APIs could not be established. Display an error message,
        // or handle the failure silently
        // ...
    }
}
```

Google APIs for Android

For a list of Google APIs for Android, see:

- https://developers.google.com/android/

Location Detection

Location Detection

- Location detection is one of the most commonly used features of smartphones
- Many apps offers location-based services, e.g.:
  - Maps, transportation and navigation
  - Location-based social networking
  - Tracking
  - News
  - Weather forecast
  - …

Location Detection

- Mobile phones can detect locations by using a variety of data sources
  1. GPS (Global Positioning System)
  2. Information of base stations (Cell ID)
     (e.g. http://opencellid.org/)
  3. Wi-Fi + IP Address
  4. Others

Location Detection

- Android provides simple APIs that return user’s current location by combining the input from different sources

Android Framework’s
Location APIs
(android.location)

Google Play Services
Location APIs
(com.google.android.gms.location)

Preferred Option

Google Play Services Location APIs

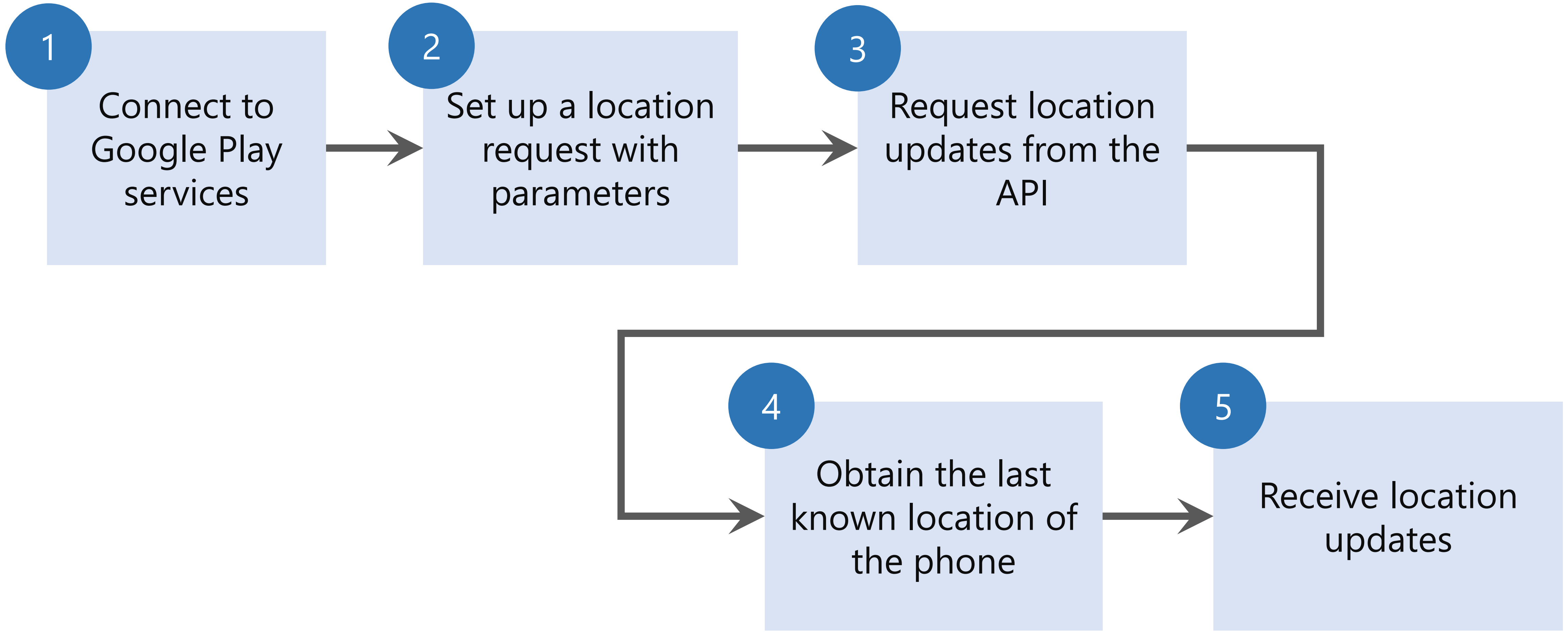
- If your app needs to use location services, you need to request permission from the user
- Two permissions:
  1. `ACCESS_COARSE_LOCATION`
     (accurate to the level of a city block)
  2. `ACCESS_FINE_LOCATION`
     (accurate to a few metres)

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

Google Play Services Location APIs

- The flow of setting up the app to use the location APIs


1
Connect to Google Play services
2
Set up a location request with parameters
3
Request location updates from the API
4
Obtain the last known location of the phone
5
Receive location updates

1. Connecting to Google Play Services

- Firstly, you need to get an instance of the Google Play services API client

```
GoogleApiClient mGoogleApiClient;

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
}
```

Adding the location service API here

NOTE: We do not enable auto-management of connection here.

1. Connecting to Google Play Services

- Connect the API client when the activity is started, and disconnect when it is stopped

```
@Override
protected void onStart() {
    super.onStart();
    if (mGoogleApiClient != null) {
        mGoogleApiClient.connect();
    }
}

@Override
protected void onStop() {
    if (mGoogleApiClient != null) {
        mGoogleApiClient.disconnect();
    }
    super.onStop();
}
```

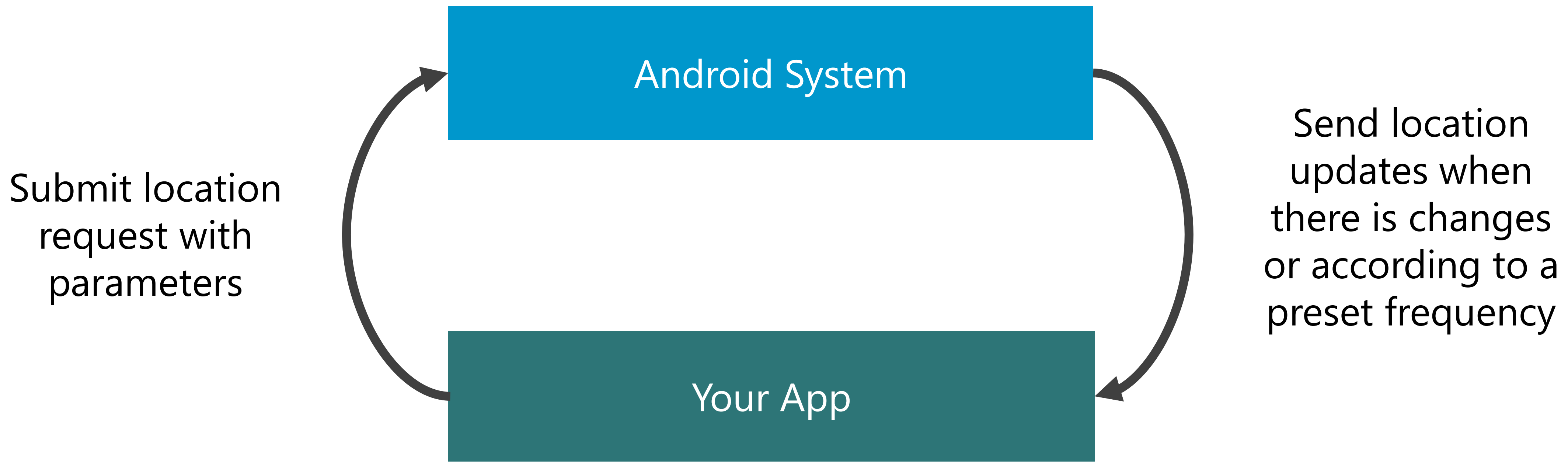
1. Connecting to Google Play Services

- To receive status updates from the API client, your activity need to implements some interfaces and callback functions

```java
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;

public class MainActivity extends Activity implements
GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener,
LocationListener {

    ...

}
```

2. Setting up a Location Request

- The Location APIs will notify your app when it has an updated information about the location of the user
- You need to submit a request to receive location updates


Android System
Your App
Submit location request with parameters
Send location updates when there is changes or according to a preset frequency

2. Setting up a Location Request

- Creating a location request:

```
protected void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(10000);
    mLocationRequest.setFastestInterval(5000);
    mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

- Interval (ms): how frequent you would like to receive updates
- Fastest Interval (ms): the fastest rate at which your app will receive updates
- Priority: Different priority settings affect power consumption
  (ref.: https://developer.android.com/training/location/receive-location-updates.html)

3. Request Location Updates

- Once the API client is connected, you can submit your location request:

```
@Override
public void onConnected(Bundle connectionHint) {
    ...
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
    ...
}
```

You need to pass to this method a location listener. If your activity has implemented the location listener interface, you can provide the reference of the current activity here.

4. Obtain the Last Known Location

- Usually it takes some time before you will receive the first location update
- You can obtain the "**last known location**" of the user first, so that you can start executing some functions in your app

```
...
@Override
public void onConnected(Bundle connectionHint) {
    mLastLocation = LocationServices.FusedLocationApi.getLastLocation(
            mGoogleApiClient);
    if (mLastLocation != null) {
        mLatitudeText.setText(String.valueOf(mLastLocation.getLatitude()));
        mLongitudeText.setText(String.valueOf(mLastLocation.getLongitude()));
    }
}
...
```

5. Receiving Location Updates

- Finally, set up the callback functions so that you can receive location updates

```
...
@Override
public void onLocationChanged(Location location) {
    mCurrentLocation = location;
    mLastUpdateTime = DateFormat.getTimeInstance().format(new Date());
    ...
    // Perform actions based on the updated location
    // e.g. update the UI, alert the user, etc.
}
...
```

Stop Receiving Location Updates

- To save power, you should stop requesting location updates when it is no longer necessary. For example:

```
@Override
protected void onPause() {
    super.onPause();
    LocationServices.FusedLocationApi.removeLocationUpdates(
            mGoogleApiClient, this);
}
```

```
@Override
public void onResume() {
    super.onResume();
    if (mGoogleApiClient.isConnected()) {
        LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
    }
}
```

Using Google Maps

Using Google Maps in Your App

- You can embed an interactive map in your app by using **Google Maps Android API**
- Google Maps functions include:
  - ➢ Provide interactive maps with 3D maps, satellite view, terrain view, road maps, etc.
  - ➢ Allow overlaying of different components, such as markers, polygons, etc.
  - ➢ Control user's view such as rotation, zoom, pan
  - ➢ Street view

Getting Started

- To embed Google Maps in your app, you need to set up your app and obtain an API key from Google
- Follow the instructions at: https://developers.google.com/maps/documentation/android/start
- Once you have a key, add it to your manifest file:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="API_KEY"/>
```

Adding a Map to Your App

- To add a map to your activity, add a fragment with `android:name` equals to "`com.google.android.gms.maps.MapFragment`":

```xml
<?xml version="1.0" encoding="utf-8"?>
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name="com.google.android.gms.maps.MapFragment"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Adding a Map to Your App

- To control the map, in the code of the activity, you need to obtain the map object:

```
public class MainActivity extends FragmentActivity implements OnMapReadyCallback {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        MapFragment mapFragment = (MapFragment) getFragmentManager()
                    .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

    @Override
    public void onMapReady(GoogleMap map) {
        map.addMarker(new MarkerOptions().position(new LatLng(0, 0)).title("Marker"));
    }
}
```

Other Maps SDKs

- Nokia HERE Map
  https://developer.here.com/
- OpenStreetMap
  http://www.openstreetmap.org/
- Baidu Map
  http://lbsyun.baidu.com/sdk/download

Geocoding & Reverse Geocoding

Geocoding & Reverse Geocoding

Geocoding

- Converting an address to a geographic location (latitude/longitude)

Reverse Geocoding

- Converting a geographic location (latitude/longitude) to an address

Both of these functions can be performed by using the `Geocoder` class in Android

Reverse Geocoding

Reverse Geocoding

- In a mobile app, you might need want to let the user choose a location on the map, and then retrieve the address of that location
  (or when you need to locate the user using GPS/Wi-Fi signals)
- To use reverse geocoding, you need a precise geolocation, therefore you should ask for the **ACCESS_FINE_LOCATION** permission
- You can use the Geocoder class to perform reverse geocoding
  (see next page)

Reference: http://developer.android.com/training/location/display-address.html

Reverse Geocoding

- The function `getFromLocation` is a blocking and may take a long time (several seconds) to response
- You should not execute this on the UI thread
- Instead, you should use one of the following methods
  - › AsyncTask
  - › Intent service (`IntentService`) with result receiver (`ResultReceiver`) or local broadcasting (`LocalBroadcastManager`)

Geocoding

- Similarly, you can obtain a geolocation by providing a address (i.e. geocoding)
- For example:

```java
String address = "The Chinese University of Hong Kong, Shatin, Hong Kong";
List<Address> list = null;

try {
    addresses = geocoder.getFromLocationName(address, 1);

} catch (IOException ioException) {
    // Catch network or other I/O problems

} catch (IllegalArgumentException illegalArgumentException) {
    // Catch invalid latitude or longitude values
}
```

Other Cloud Services

Cloud Services & APIs

- Facebook
  https://developers.facebook.com/
- Instagram
  https://www.instagram.com/developer/
- WeChat
  https://open.weixin.qq.com/
- Baidu
  http://developer.baidu.com/
- YouTube
  https://www.youtube.com/yt/dev/
- IBM Developer Cloud
  https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/

End of Lecture 10