Cloud to Device Messaging Framework

丸山不二夫
早稲田大学／はこだて未来大学
@maruyama097

Cloud to Device Messaging C2DMとは何か

- 開発者が、サーバからデータを、Androidデバイス上のアプリケーションに送るのを助けるサービス。
- 単なるPush型のNotificationサービスではなく、Androidのシステムと統合されている。
- Android上のアプリを、メッセージを受け取るために走らせておく必要はない。
- メッセージが届いた時、Androidは、Intent Broadcastのメカニズムを用いて、アプリを起動する。

Google IO 2010 でのデビュー[動画]

DEMO

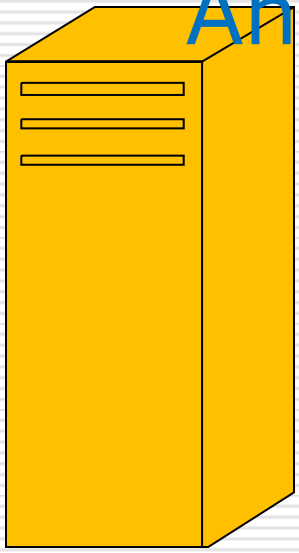
chrometophone

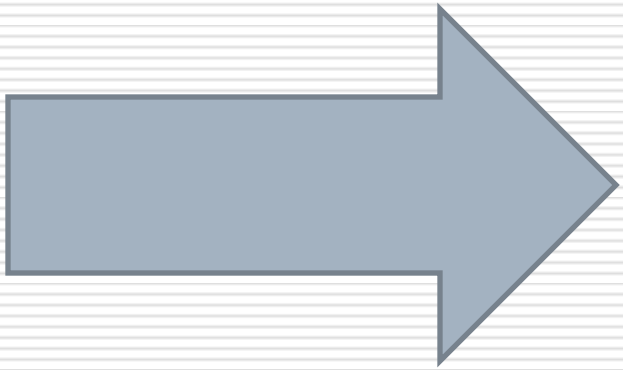
http://code.google.com/p/chrometophone/

C2DMの概略

基本的な情報の流れ

開発者が設置したサーバから
Androidにメッセージを送る



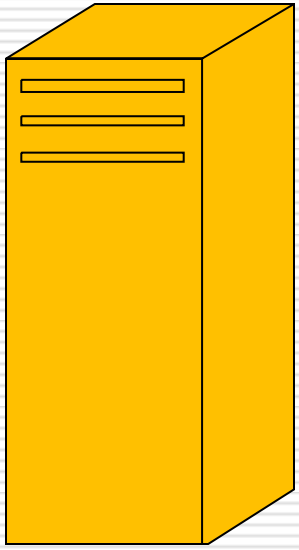


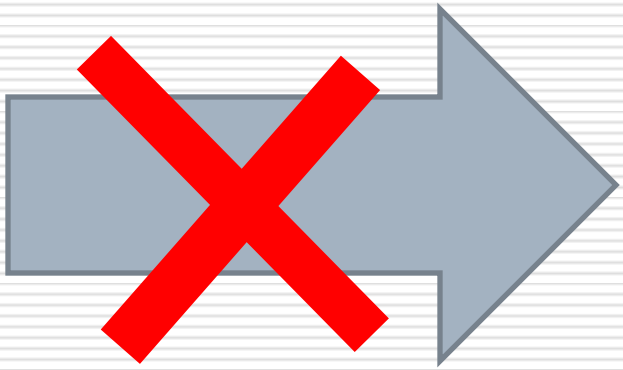
アプリケーション・サーバ

モバイル・デバイス

ただし、……

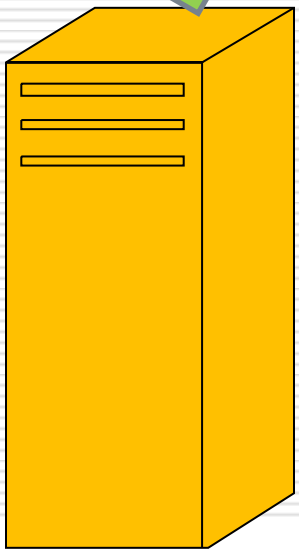
開発者のサーバから直接
Androidにメッセージを
送るわけではない





なぜなら、モバイル・デバイスは、繋がらないこともあれば、通信の遅延が大きい時もある。

Google
C2DMサーバ



C2DMでの実際の情報の流れ

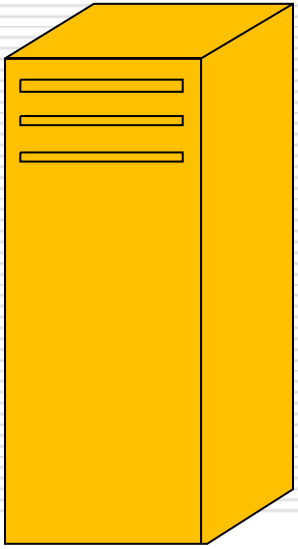
C2DMサーバが、メッセージを一時的に蓄え、Androidへの配送に責任を持つ。

C2DMサーバは、どのようにして配送先を知るのか？

GoogleのC2DMサーバは、そのままでは、メッセージの配送先を知らない。メッセージの受け取り手が、C2DMサーバに対して、受け取り手であるという登録を行う必要がある。

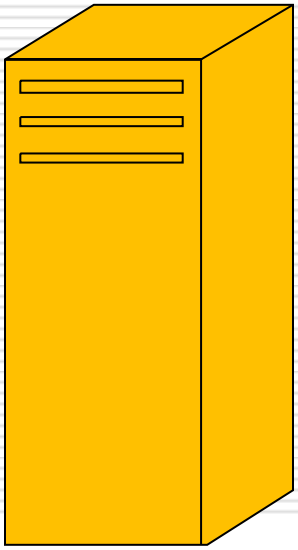
Google
C2DMサーバ

1．登録



まず、Androidが、C2DMサーバに対して、メッセージの受け取り手であるという登録を行う。

デバイスの指定だけでは不十分で、どのアプリがメッセージを受け取るのかC2DMサーバに知らせる必要がある。デバイス中のアプリは、次のようなIDを持っている。

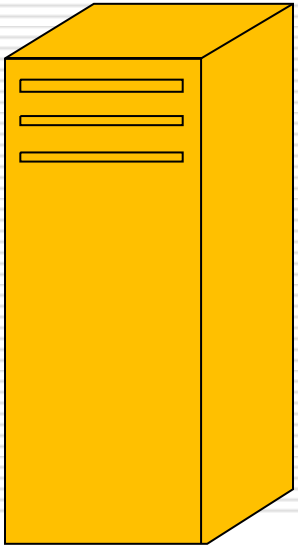


Sender ID

Application ID

Google
C2DMサーバ

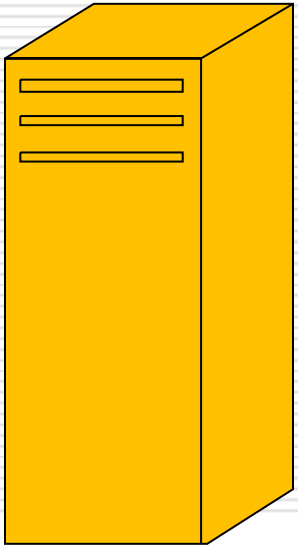
2. 登録 ID



C2DMサーバは、
登録を受け付けると、
Androidに対して、
登録IDを返す。

Google
C2DMサーバ

アプリケーション・
サーバは、受け取った
登録IDをきちんと記憶
しておかなくてはならない



Androidは、C2DM
サーバから受け取った
登録IDを、アプリケー
ション・サーバに
伝える。

3. 登録ID

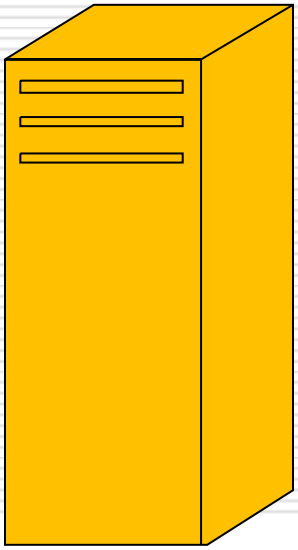
こんな感じ

Google
C2DMサーバ

三者の通信は、この
Registration IDに
基づいて行われる。

1. Register

2. Registrtion ID



3. Registrtion ID

アプリケーション・サーバ

モバイル・デバイス

サーバは、C2DMサーバに
どのようなメッセージを送るのか？

メッセージの送信　ClientLogin token

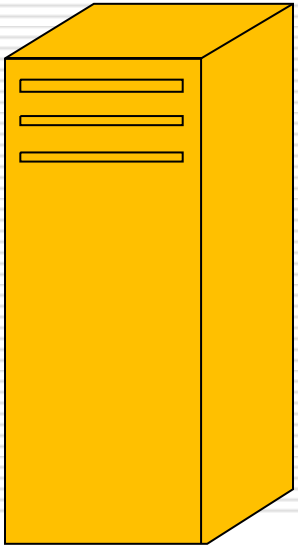
- アプリケーション・サーバがメッセージを送るためには、もう一つ用意しておくべきことがある。
- ClientLogin authorization tokenである。これは、そのアプリの為に、開発者があらかじめアプリケーション・サーバにセットアップしておくべきもので、デバイスにメッセージを送る際に利用される。

Google
C2DMサーバ

POST

ClientLogin Auth

トークン＋メッセージ

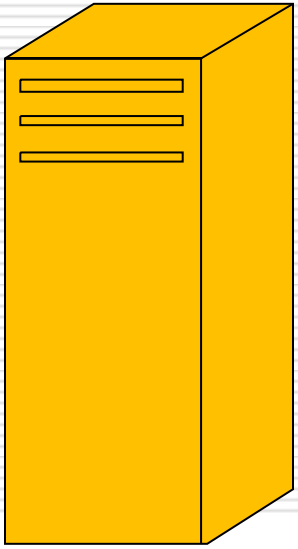


アプリケーション・サーバ

モバイル・デバイス

Google
C2DMサーバへの
メッセージの送信

POST https://android.apis.google.com/
c2dm/send
registration_id
collapse_key
data.<key>
ClientLogin Auth



アプリケーション・サーバからのPOSTに含まれるフィールド

- **registration_id**
  端末上のAndroidアプリから取得したregistration ID。必須。
- **collapse_key**
  デバイスがオフラインの時など、最後のメッセージだけがクライアントに送られるように、類似したメッセージのグループを折りたたむために使う、任意の文字列。端末がオンラインに復帰した時、沢山のメッセージが送られないためのもの。必須。メッセージの順序の保証はないので、本当に「最後」のメッセージではないかもしれないことに注意

アプリケーション・サーバからのPOSTに含まれる情報

- **data.<key>**
  key/valueペアで表現されたデータ。
  <key>を持ったアプリケーション・データとして、Intentの中に含まれることになる。
  key/valueペアの数に制限はないが、メッセージの全体のサイズには、制限がある。

アプリケーション・サーバからのPOSTに含まれる情報

- **delay_while_idle**
  もし含まれていれば、デバイスが動いていないとき、メッセージをすぐには送らないことを示す。サーバは、デバイスがアクティブになるのを待って、それぞれの**collapse_key**毎に、その値の、最後のメッセージを送信する。
- **Authorization**:
  GoogleLogin auth=[AUTH_TOKEN]
  ClientLogin Authトークン。

C2DMサーバは、
どのような働きをするのか？

C2DMサーバの働き

1. C2DMサーバは、アプリケーション・サーバからおくられたメッセージを受け取る。
2. C2DMサーバは、受け取り手のAndroidデバイスが立ち上がっていない場合には、メッセージをキューに入れて蓄えておく。
3. Androidデバイスがオンラインになったら、C2DMサーバは、メッセージを送る。

Google
C2DMサーバ

Send Message

モバイル・デバイス

Androidは、どのようにC2DMからのメッセージを処理するのか？

Androidシステムの C2DMサーバからのメッセージ受信

- Android側では、システムは、特定のアプリに対して、適当なパーミッション設定をし、そのターゲットのアプリだけがメッセージを取得できるようにしておかなければならない。
- システムは、C2DMサーバから送られたメッセージを受け取り、メッセージのpayloadから、生のkey/valueペアを受け取る。

Androidシステムの行う Intentの作成と配信

- システムは、C2DMサーバから受け取ったkey/valueペアの情報を、**com.google.android.c2dm.intent.RECEIVE Intent**のExtrasに詰める。
- システムは、Intent Broadcastを使って、メッセージをブロードキャストする。
- これで、アプリは立ち上がる。メッセージを受け取るために事前に動いている必要はない。

Androidアプリの行う処理

- Androidアプリは、RECEIVE Intentから、keyを用いて生のデータを取り出し、データを処理する。

Androidアプリ側での処理

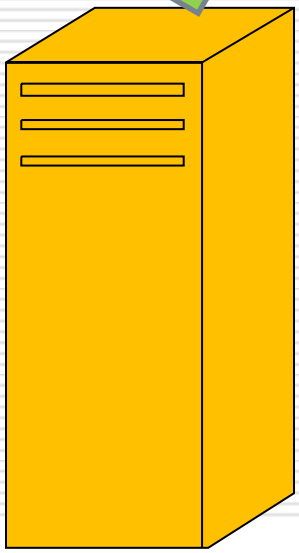
```
public final void onHandleIntent(Intent intent) {
  try {
    Context context = getApplicationContext();
    if (intent.getAction().equals(
     "com.google.android.c2dm.intent.REGISTRATION"))
     {
      handleRegistration(context, intent);
     } else if (intent.getAction().equals(
     "com.google.android.c2dm.intent.RECEIVE")) {
      onMessage(context, intent);
```

AndroidアプリがC2DMからメッセージを受け取るのは、登録IDを受け取る場合と、アプリケーション・サーバから送られたメッセージを受け取る場合の二つがある。

chrometophoneの構造を見る

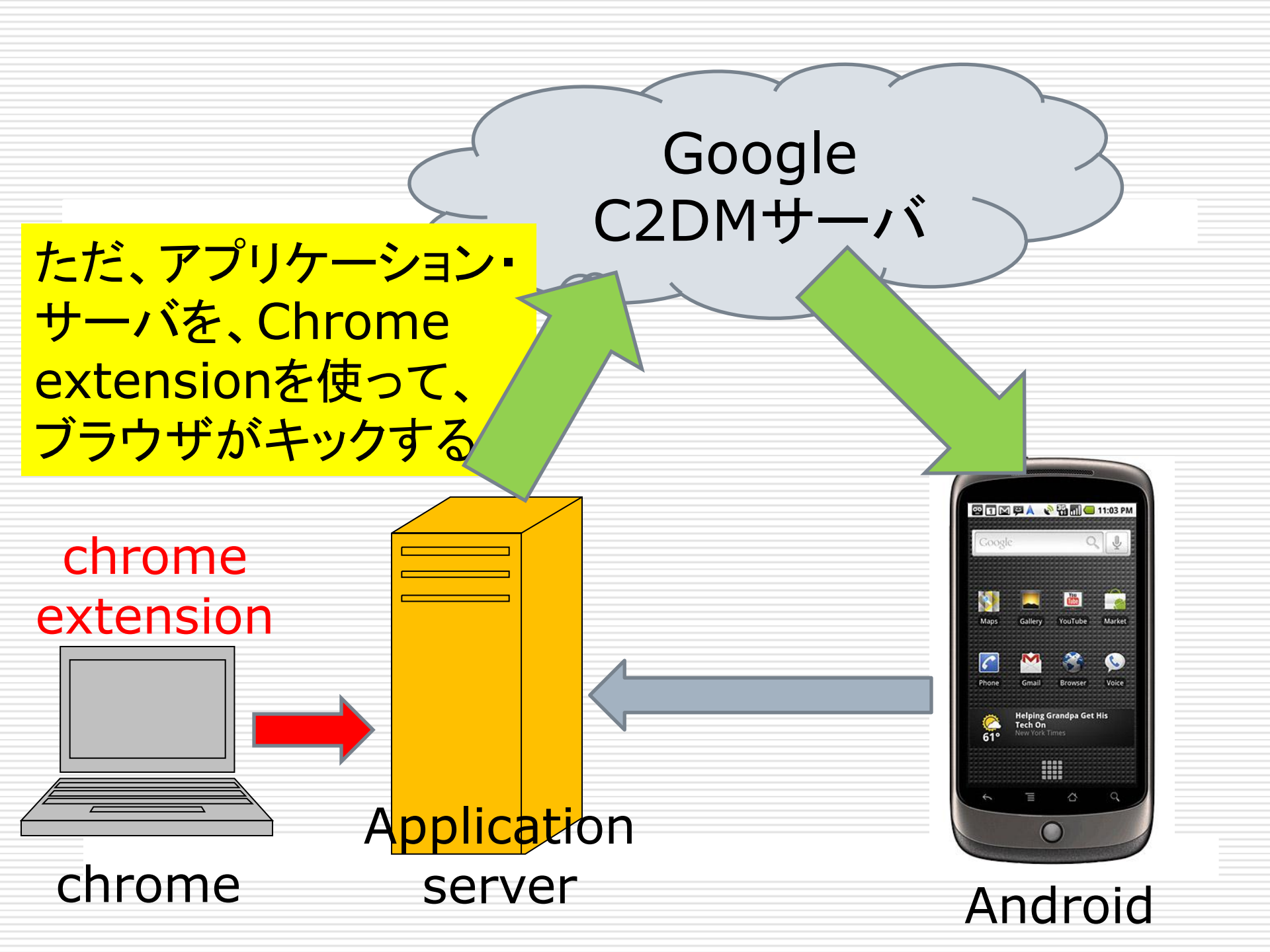
ここでは、アプリケーション・サーバの働きを中心に、chrometophoneのサンプルを分析しよう。

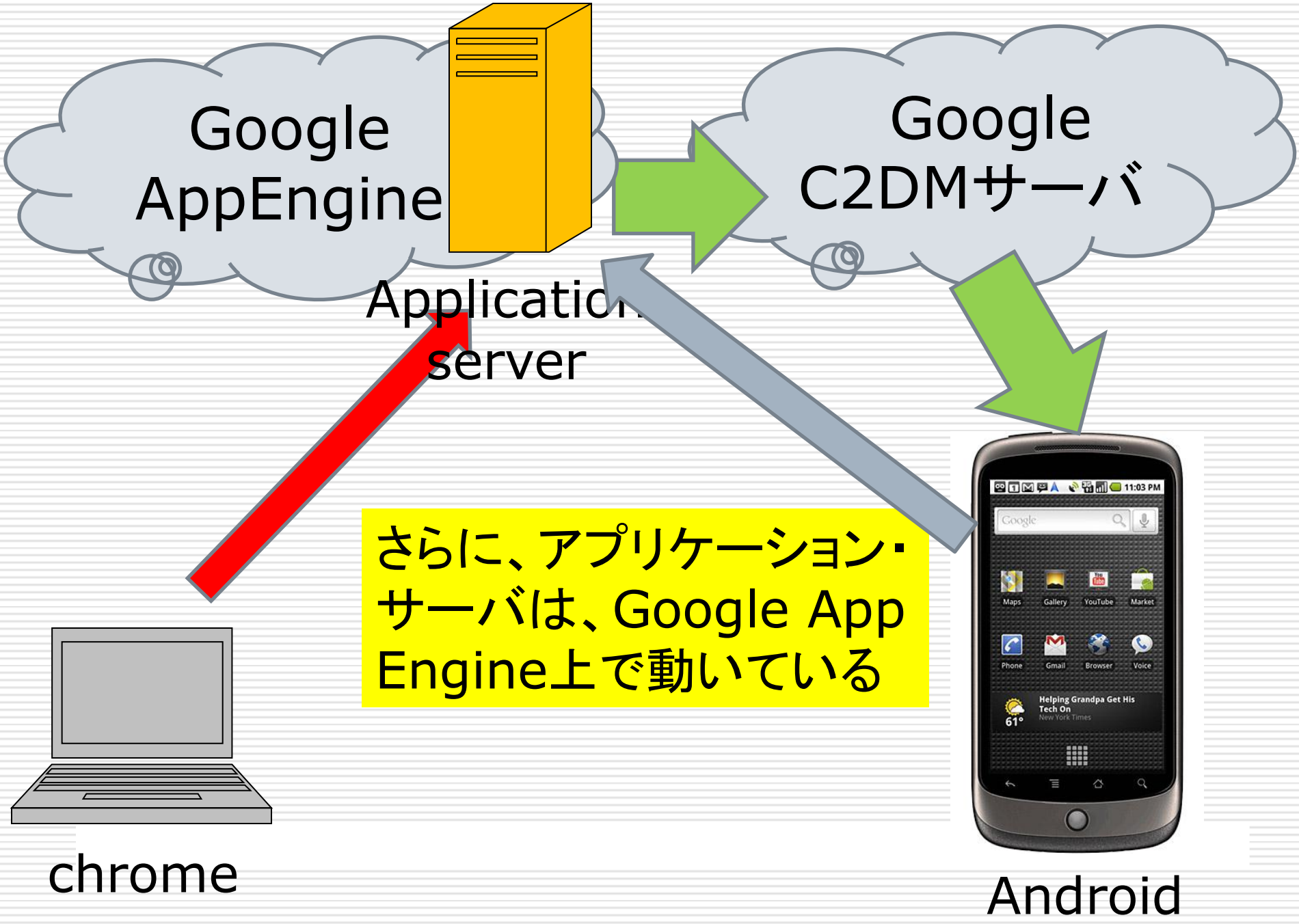
Google
C2DMサーバ



chrometophone の基本的な構造

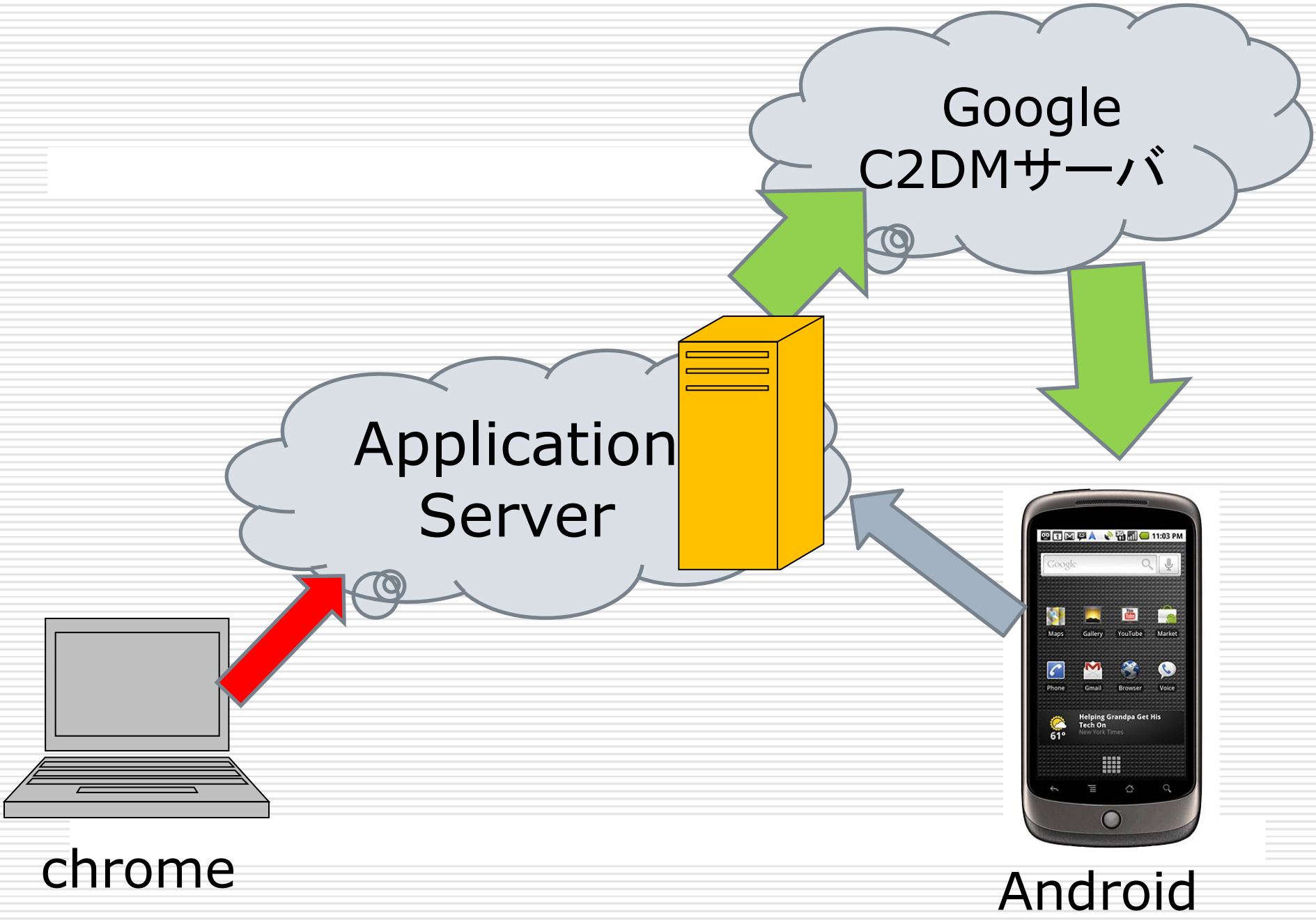
chrometophone でも、基本的には、C2DMのこうした構造には変わりない


Google
C2DMサーバ
ただ、アプリケーション・サーバを、Chrome extensionを使って、ブラウザがキックする
chrome extension
chrome
Application server
Android

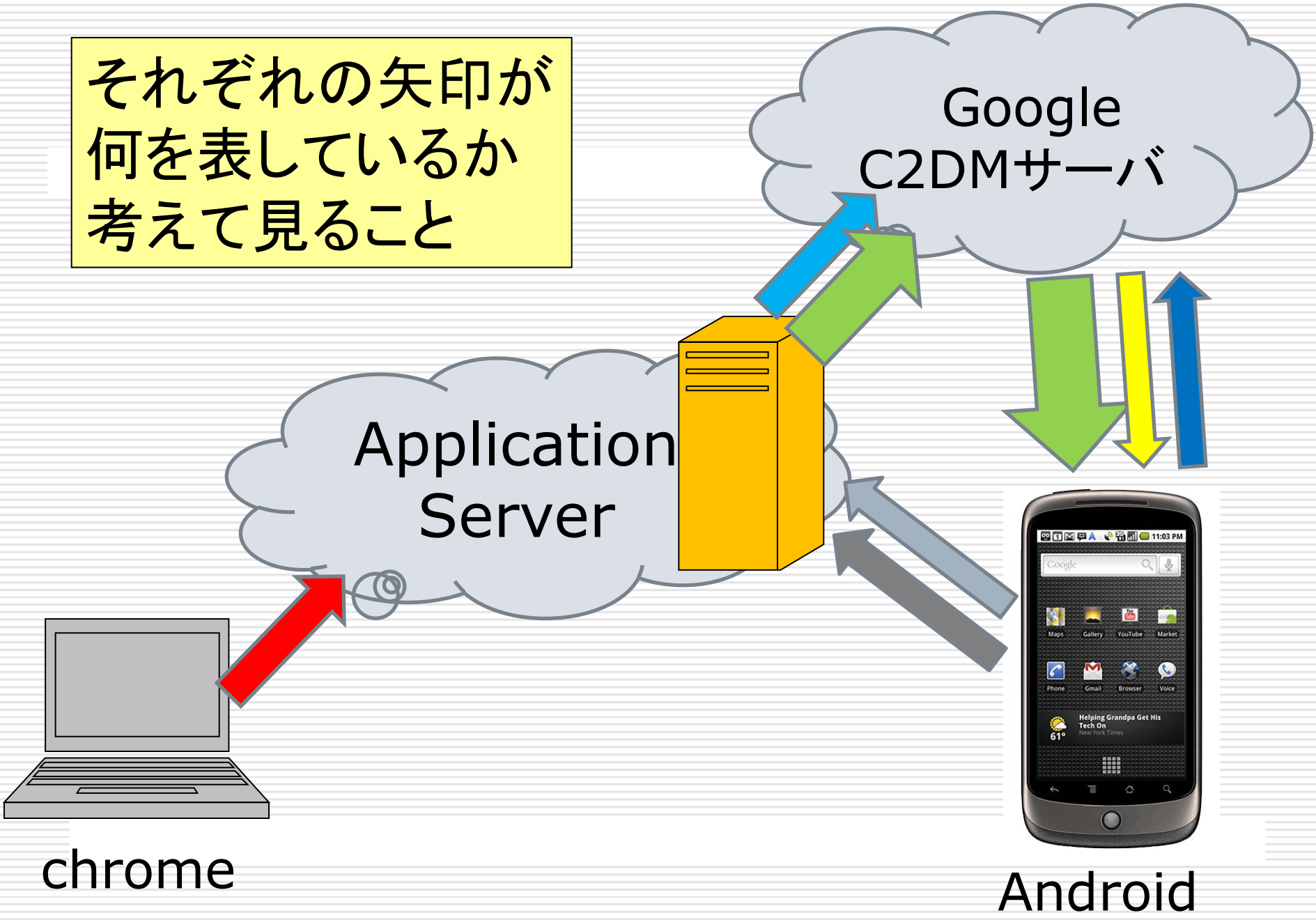

Google
AppEngine
Google
C2DMサーバ
Application
server
さらに、アプリケーション・サーバは、Google App Engine上で動いている
chrome
Android

chrometophoneでのアプリケーション・サーバの働き

ソースの**chrometophone-read-only/appengine/war/WEB-INF/web.xml**を見れば、アプリケーション・サーバの働きがよく分かる。アプリケーション・サーバは、ここでは、すべてServletとして構成されている。


Google
C2DMサーバ
Application
Server
chrome
Android

それぞれの矢印が
何を表しているか
考えて見ること



chrometophoneアプリケーション・サーバを構成するServletとURL

- **RegisterServlet** /register
  - 登録IDの登録・記録
- **UnregisterServlet** /unregister
  - 登録IDの抹消
- **SendServlet** /send
  - C2DMサーバへのデータ送信
- **AuthServlet** /signin, /signout
  - C2DMサーバへの認証
- **dataMessagingServlet** /tasks/c2dm
  - Chrome Extensionとの通信

RegisterServletの doPostメソッドの一部

```
User user = checkUser(req, resp, true);
if (user != null) {
   Key key = KeyFactory.createKey(DeviceInfo.class.getSimpleName(),
                                   user.getEmail());
   DeviceInfo device = new DeviceInfo(key, deviceRegistrationID);
   // Context-shared PMF.
   PersistenceManager pm =  C2DMessaging.getPMF(getServletContext()).
                                   getPersistenceManager();
   try {
      pm.makePersistent(device);
      resp.getWriter().println(OK_STATUS);
   } catch (Exception e) {
      resp.setStatus(500);
      resp.getWriter().println(ERROR_STATUS + " (Error registering device)");
      log.warning("Error registering device.");
    } finally {….}
}
```

RegistrationIDは、データベースに格納されている。

SendServletの doPostメソッドの一部

```
// Send push message to phone
C2DMessaging push = C2DMessaging.get(getServletContext());
boolean res = false;
String collapseKey = "" + url.hashCode();
if (deviceInfo.getDebug()) {
      res = push.sendNoRetry(
            deviceInfo.getDeviceRegistrationID(),
            collapseKey,
            "url", url,
            "title", title,
            "sel", sel,
             "debug", "1");
} else {……}
```

chrometophoneでの Chrome Extensionの働き

manifest.json

```
{
  "name": "__MSG_app_name__",
  "description": "__MSG_app_desc__",
  "version": "2.0.0",
  "default_locale": "en",
  "icons": { "128": "icon_128.png" },
  "options_page": "help.html",
  "browser_action": {
    "default_title": "__MSG_app_name__",
    "default_icon": "icon_19.png",
    "default_popup": "popup.html"
  },
  "permissions": [
    "tabs", "http://*/*", "https://*/*"
  ],
  "content_scripts": [
    {
      "matches": ["http://*/*", "https://*/*"],
      "js": ["content_script.js"]
    }
  ]
}
```

ブラウザー右上の
ボタンを押した時に
popupするウィンドウ
chrometophoneの
動作は、基本的に
この中で定義されている

content_script.js

```
var pageInfo = {
  "url": document.location.href,
  "title": document.title,
  "selection": window.getSelection().toString()
};
// URL overrides
if (pageInfo.url.match(/^http[s]?:¥/¥/maps¥.google¥./) ||
    pageInfo.url.match(/^http[s]?:¥/¥/www¥.google¥.[a-
z]{2,3}(¥.[a-z]{2})¥/maps/)) {
  var link = document.getElementById('link');
  if (link && link.href) {
    pageInfo.url = link.href;
  }
}          //  どこにメッセージを送っているのか考えて見ること
chrome.extension.connect().postMessage(pageInfo);
```

popup.html（一部）

```
<script type="text/javascript">
var apiVersion = 4;      //  ここに送っている
var baseUrl = 'https://chrometophone.appspot.com';
var sendUrl = baseUrl + '/send?ver=' + apiVersion;;

var signInUrl = baseUrl + '/signin?extret=' +
encodeURIComponent(chrome.extension.getURL('help.html')
) + '%23signed_in&ver=' + apiVersion;

var signOutUrl = baseUrl + '/signout?extret=' +
encodeURIComponent(chrome.extension.getURL('signed_ou
t.html')) + '&ver=' + apiVersion;

var req = new XMLHttpRequest();
```