GeoWeb Services and Open Web Mapping

1. Background

For many years, most of the digital geographic datasets were confined for personal use on desktop-based personal computers (PCs) and therefore could not be easily shared with other organizations or user groups. GIS professional would access data from their own computers that were often connected to a central computing server located in the office. Dedicated software meant for GIS was required to view or manipulate the data, effectively limiting and narrowing the other users that could benefit from the data. When people started using the Internet in the mid-1990s, they began thinking about how maps and other geographic information could be shared across heterogeneous networks, both within the office and also with the public for maximizing the data usage. The early version of web mapping tools such as Map Server and Esri ArcIMS, were really pixelated, slow, and clunky by today's standards but then these initial interactive web maps were revolutionary at the time. Further, the problem was compounded as these dynamically drawn web maps had trouble with speed and scalability.

With the rise of Google Maps, there came out mapping tools that enables serving of “pre-seeded” tiled map images using a technique known as Asynchronous JavaScript and XML (AJAX) that eliminated the ubiquitous and annoying page blink that occurred after any navigation action in earlier web maps. Web maps started using cartographic techniques to make the most attractive maps. This lead to birth of beautiful, fast, and detailed “web 2.0” basemaps that are prevalent today on Google, Microsoft Bing, OpenStreetMap, and other popular websites.

2. Importance of Web Services

All the web mapping applications today are made possible because of web services. A web service can be think of as a focused task that a specialized computer (the server) knows how to do and allows other computers to invoke. We work with the web services as follows:

1. First invoke the web service by making a request from an application (the client). To make this request, we usually use HTTP, a standard protocol that web browsers use for communicating between clients and servers. The request contains structured pieces of information called parameters. These give specific instructions about how the task should be executed.
2. The server then reads the request and runs its web service code, considering all the parameters while doing so. This produces a response, which is usually a string of information or an image.
3. The server sends you the response, and our application uses it.

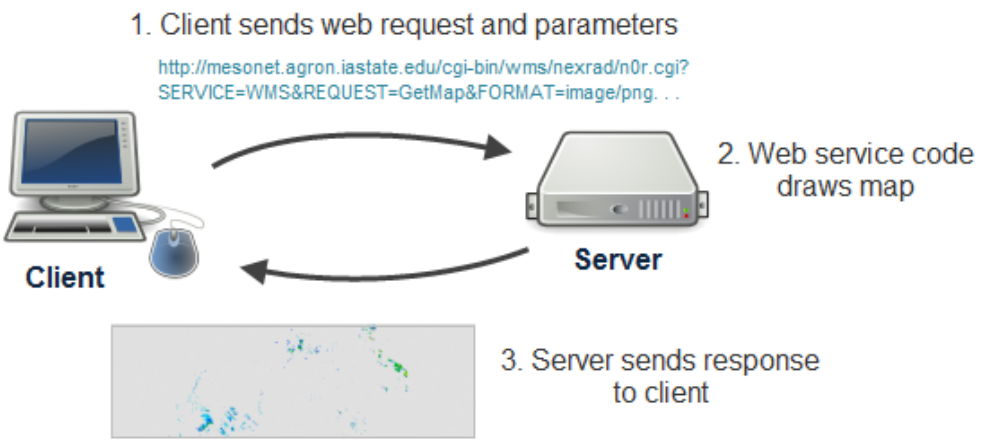

Figure: Example request and response flow of a web service that draws maps.

Let's assume that you've identified the URL of a web service out there somewhere that draws maps. You make a request by constructing a URL (http://...) containing the address of the web service and various parameters for the map, such as the format of the image you want to receive (JPG, PNG, etc.), bounding box (coordinates defining the geographic area you want to see mapped), and map scale. You paste this into your browser's address bar and the server sends you a response containing the map image you requested. Here's an example of just such a request, using a radar map of the US. First, see if you can identify some of the parameters in the URL. Then take a guess at what the response will look like when you get it back. Then click the link to invoke the web service and get the response:

```
http://mesonet.agron.iastate.edu/cgi-
bin/wms/nexrad/n0r.cgi?SERVICE=WMS&REQUEST=GetMap&FORMAT=image/png&TRANSPARENT=TRUE&ST
YLES=&VERSION=1.3.0&LAYERS=nexrad-n0r&WIDTH=877&HEIGHT=276&CRS=EPSG:900913&BBOX=-
15252263.28954773,2902486.4758432545,-6671748.242369267,5602853.811101243
```

Examined the URL of this request and you will notice the parameters indicating the width and height of the image, the image format, the image background transparency, and the bounding coordinates of the map to be drawn. These parameters provide specific details about how the web service should run its map drawing code. You see these parameters reflected in the response image sent back to your browser when you clicked the link. In a future lesson, you'll learn about more of the parameters in the request above. Not every web requests invoke web service code. Some web requests just returns a file. This is how tiled maps work, and this is why they are so fast. Examine the following request for a specific zoom level, row, and column of a tile:

```
https://a.tile.openstreetmap.org/15/11068/19742.png
```

The above URL request drills down into the server's folder structure and returns you the requested PNG image as a response. No special code ran on the server other than the basic file retrieval, therefore you could argue that a web service was not invoked. However, this type of simple web request is also an important part of many web maps. Also note that, not all web services use the same format of URL and parameters.

3. Open Specification, Standards and Data

Open specifications are documented and mutually agreed-upon patterns of how software and digital data should behave in order to be interoperable between systems. For example, HTTP (hypertext transfer protocol) is based on a specification defining how web servers and web browsers should communicate in order to exchange information. Without open specifications, you would not be reading this web page. There are two types of open specifications:

- Open data formats – Most GIS data formats are open in the sense that the way they are constructed is fully documented, and various GIS programs can read and write them.

  KML and GeoJSON are some examples of GIS data formats that you can create just by writing a text document in a particular way. Most raster formats such as JPEG or PNG are also open. The shapefile is one of the most common formats for exchanging vector GIS data, because Esri has openly documented how to create a shapefile and relinquished any legal restrictions on creating shapefiles. In contrast, an example of a closed data format is the Esri file geodatabase, because Esri has not openly documented how to create a file geodatabase without using Esri tools.

- Open specifications for web map services – There have been several efforts to openly document patterns that GIS web services should use when communicating with clients. The Open Geospatial Consortium (OGC) has created several of these specifications, the most popular of which is the Web Map Service (WMS).

The Open data that has been made available to the public free of charge and bereft of most copyright restrictions. These data may be shared by government entities, researchers, not-for-profit organizations, or ordinary citizens contributing to online projects. For example, Data.gov is a popular website offering datasets collected by the US government. OpenStreetMap is an online map consisting of voluntary contributions in the style of Wikipedia.

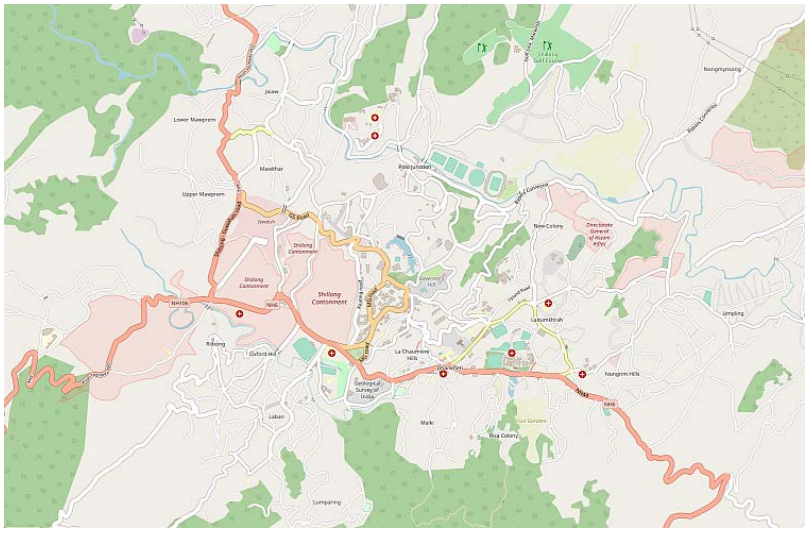

Figure: Detailed OpenStreetMap data in Shillong

Open datasets are often rich and exciting to include in web maps, but there are some precautionary measures you should follow in order to be successful with using them. First, be aware that even though the data is free, you are often still required to provide attribution describing where you obtained the data. You may also be restricted from redistributing the data in any way that requires a fee. When you use an open dataset, you are responsible to carefully research and adhere to any attribution requirements. You should also make an effort to verify the data quality by examining any accompanying metadata, researching the sources and collection methods of the data, and scrutinizing the data itself.

4. The web mapping architecture

Several different physical machines can be coupled to create, serve, and use a web map. These are often depicted in diagrams as separate levels, or tiers of architecture. In this course, you'll likely use just one machine to play all these roles; however, it's important to understand how the tiers fit together.

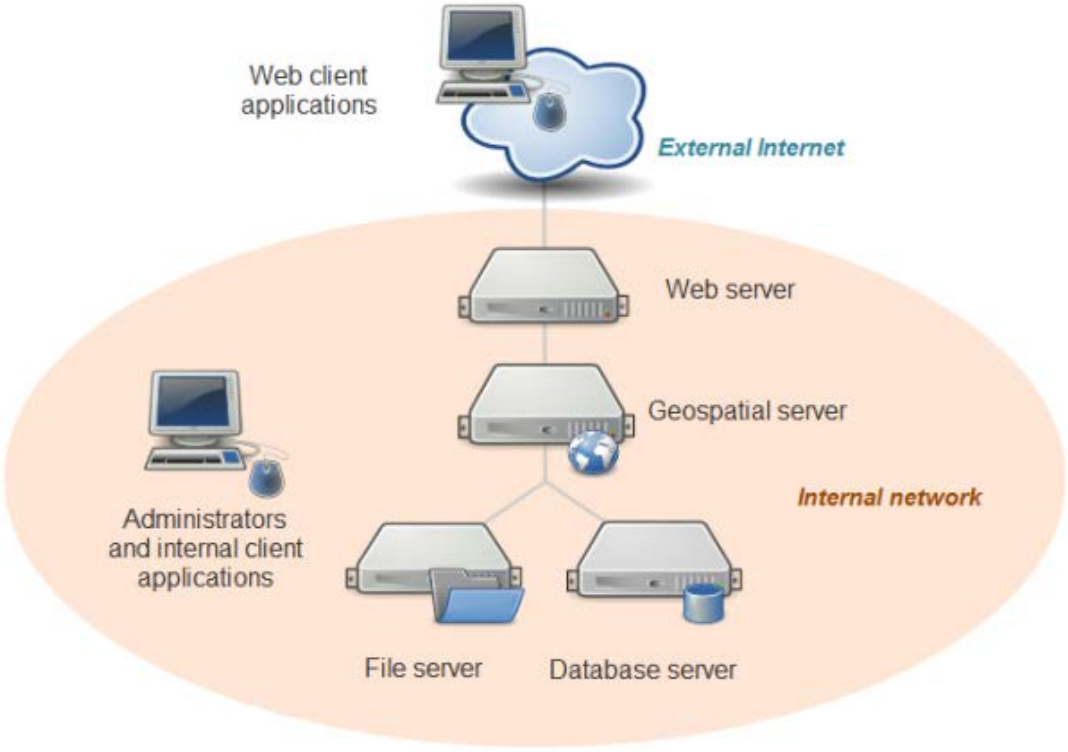

Figure: The web mapping architecture

The architecture will have the followings:

- The desktop PCs that are used by administrators and internal client applications. These machines will also be used to prepare data, author maps, and sometimes administer the other machines.
- Database and/or file server holding all of your GIS data. This machine might be equipped with redundant storage mechanisms and regular backup scripts that prevent the loss of data.
- Geospatial web services server that has specialized software and processing power for drawing maps, responding to feature queries, and performing GIS analysis operations.

- Web server that acts as a web entry point into your organization's network. This is also called a proxy server. It is protected by firewalls that shield malicious traffic into your internal network. It's also a place where you can put web application code (such as HTML and JavaScript files) for your web maps.
- Client applications that use the web map. These can be apps that run on your desktop workstation or they could be mobile apps. The clients may be based within your network, or you may allow them to come from outside your network. All clients must be able to make web requests through HTTP, and any client coming from outside your network must have an Internet connection. It's important to design for different browser versions, screen sizes, ambient lighting, and so forth.

5. Elements of a web map

Most web maps break apart the layers into groups that are handled very differently. Layers whose sole purpose are to provide geographic context are grouped together and brought into the map as a single tiled basemap. In contrast, thematic layers (the layers that are the focus of the map) are brought in as one or more separate web services and placed on top of the basemap. You might additionally decide to include a set of interactive elements such as popups, charts, analysis tools, and so forth.

- Basemaps: A basemap provides geographic context for your map. In other words, it is usually not the main reason people look at your map, but your map would be difficult to interpret without it. The most common basemaps you've used online are vector road maps and remotely sensed imagery. Although a basemap may consist of many sublayers (such as roads, lakes, buildings, and so forth), these are often fused together into a rasterized set of tiled images and treated as a single layer in your web map. These tiled maps consist of often thousands or millions of pre-drawn images that are saved on the server and passed out to web browsers as people pan around the map.

- Thematic Layers: Thematic layers (also known as business or operational layers) go on top of the basemap. They're the reason people are coming to visit your map. If placed on the basemap, they might not be of interest to everybody, but when placed on your focused web map, they are the main layer of interest. Like basemaps, thematic layers are sometimes displayed with tiles; however, this may not always be possible due to the rapidly changing nature of some data. For example, if you need to display the real time positions of police vans, you cannot rely on predrawn tiles and must use some other way to draw the data. There are various web services such as WMS that are designed to draw maps on the fly in this way. You can use these for your thematic layers. Another option is to query the server for all the data and use the browser to draw it. This approach lends itself well to interactive elements such as popups. Thematic layers work together with basemap layers to form an effective web map. Interestingly, the thematic layer is not always the top one.

- Interactive Maps: Web maps are often equipped with interactive elements that can help you learn more about the layers in the map. These can be informational popups that appear when you click a feature, charts and graphs that are drawn in a separate part of the page, slider bars that adjust the

time slice of data displayed in the map, and so forth. Some web maps allow you to edit GIS data in real time, or submit a geoprocessing job to the server and see the response drawn on the screen. These elements make the map come alive. The key to making an effective web map is to include the interactive elements that are most useful to your audience, without overwhelming them with options or making the tasks too complicated. Interactive elements are the part of your web map that requires the most custom programming. The amount of interactivity you have the freedom to add may be strongly correlated with the amount of JavaScript programming that you know how to do. Open web mapping APIs such as OpenLayers and Leaflet provide convenient methods for doing some of the most common things, such as opening a popup.

6. Dynamically drawn map services

There are various ways to get a map to show up in someone's web browser. One way is for the server to send predrawn map images (or image tiles) to the browser, another way is for the server to send a bunch of text-based geographic coordinates and attributes that the browser draws, and a third way is to draw the map on the server at the time of request and send the browser the completed map image. It's sometimes called a dynamic map service because, being drawn on the fly, it reflects the most recent picture of the data. Contrast this with the image tile approach, which represents a static picture of the data taken at one point in time (when all the tiles were generated).

- Advantages of dynamically drawn map services: Because dynamically drawn map services retrieve the data and draw it at the time of request, they are useful for gaining a situational awareness of the most recent state of the data. They may be the only reasonable way to depict many features that are changing at the same time (such as the positions of each vehicle in a large fleet). Dynamically drawn maps may also be the best solution at large scales where tiled maps become too cumbersome to generate, store, or maintain. Maps dynamically drawn through WMS allow you to apply a wide range of symbols and styles using a concept called Styled Layer Descriptors (SLDs)

- Disadvantages of dynamically drawn map services: Waiting for the server to draw the map can be a slow painful experience, especially if there are many layers to render. A wait time of 2 - 3 seconds that may be considered acceptable on the desktop may not be tolerated by edgy web map users who are neither knowledgeable nor sympathetic to the type of back end technology being used. People now expect every map to perform as fast as Google Maps and, without using tiles, this can be difficult to achieve. Dynamically drawn services are also much more susceptible to overload than tiled services if you have many users deciding they want to see the map at the same time.

7. OGC Open specifications for web map services

The OGC is an industry consortium that develops open specifications for spatial data and web services. The OGC is comprised of representatives from private companies, government organizations, NGOs, and universities.

OGC has produced a series of specifications for GIS web services named in the format Web__Service (sometimes abbreviated as W*S). For example:

- Web Map Services (WMS) return a rasterized image of a map drawn by the server. This doesn't allow much complex analysis, but it is so useful for visualization that WMS has become the most widely implemented and supported of the W*S services.
  The basic WMS draws the map on the fly, but additions to the WMS specification provide syntax for requesting predrawn image tiles from the server. To get started, we'll just work with the dynamically drawn WMS.
- Web Feature Services (WFS) return vector geometries and attributes allowing for spatial analysis and editing. The vector features are communicated in XML using Geography Markup Language (GML), itself an open specification defined by the OGC for transferring vector feature data.
- Web Coverage Services (WCS) return blocks of data called coverages (not to be confused with the old vector Arc/INFO coverages) that are commonly used for raster display and analysis.
- Web Processing Services (WPS) allow a user to invoke geoprocessing operations on the server.

8. Understanding web mapping API

An API (application programming interface) is a framework that you can use to write a program. It provides a set of classes and functions that help you avoid writing all the low-level code to perform specific actions. For example, web mapping APIs typically include classes for maps and layers so that you don't have to write all the low-level code for displaying an interactive map image and drawing a new layer on it. Instead, you can just create a new map object, create a new layer object, and call some method such as `layer.addTo(map)`. The API abstracts the complexity of the task and makes it easy for you to focus on the mapping aspects of your application, rather than spending time on the low-level logistics. APIs designed specifically for the purpose of making web maps include OpenLayers, Leaflet, the Google Maps API, and the ArcGIS API for JavaScript. The latter two are even more specific in that they are designed around particular proprietary platforms.

- Choosing a web mapping API: The selection of an API is often tightly coupled with the decision of a platform and programming language. These two factors affect the APIs available to you. For example, if you know that your application is required to run on Android tablets, you first need to decide whether you are going to build a full-fledged native app (in other words, one that is available in Google Play and has access to the device hardware such as the camera) or one that simply runs in a web browser on the Android tablet. Developing a native app most likely means that you'll be using Java, while developing a browser-based app allows more flexibility and can be done with JavaScript and HTML, perhaps employing an API that's designed to be mobile-friendly (in other words, it supports touch gestures, resizes to device width, and so forth).

9. Open Source web mapping APIs

Below list some examples of Open Source web mapping APIs for building browser-based apps with HTML and JavaScript.

- OpenLayers: OpenLayers is a mature and richly featured JavaScript API for building web map applications. It has an extensive collection of documentation and samples. One of nicest things about OpenLayers is the large developer community using the API.

- Leaflet: Leaflet is designed to be lightweight, mobile-friendly, and easy to get started with. Leaflet places heavy emphasis on the use of tiled maps and client-side vector graphics drawn from sources such as GeoJSON. For basic maps that use these layer types, Leaflet is an excellent choice that has already endeared itself to many GIS developers. Leaflet contains a full API reference.

- D3: D3 is a data visualization library that is frequently used for charting but also contains many map examples. It binds data elements to the page's document object model (DOM), allowing for interesting and flexible data animations and transitions. D3 is a nice option for composing a web app with interactive maps and charts.

- Polymaps: Polymaps is simple mapping API primarily designed for mashing up map tiles with vector features drawn from GeoJSON and other sources. Polymaps can transform and overlay a raster image onto an existing tile set. The k-means clustering example demonstrates a unique ability of the Polymaps API to generalize a large set of points on the fly.

----------x----------