MediaGrid – Distributed Media-Sharing Platform

Dan Staples, lead developer

Design Goals

As advancing consumer technologies further citizens' abilities to communicate and share at ever expanding rates, government surveillance and censorship capabilities have similarly kept up. A recent dramatic example of this came during the anti-Mubarak protests in Egypt, in which internet and cell phone services were shut down by the government in order to curtail activists' abilities to communicate and plan protests. The dangers of censorship have been seen in the US, too. In August 2011, the Bay Area Rapid Transit (BART) authorities in San Francisco shut down cell phone towers during a protest against police violence. The criminalization of recording police activities has seen a disturbing increase in the US in the past few years.

This has been the inspiration behind the MediaGrid project, envisioned as a censorship-resistant, distributed wireless infrastructure allowing citizens to share real-time video and photo documentation at public demonstrations. The design goals of the MediaGrid project are as follows:

- Create a wireless mesh network that can be accessed by any wifi-enabled devices, particularly smartphones.
- Allow for uploading of photo and video files onto a distributed filesystem.
- The mesh grid must be cheap and easy to configure.
- The mesh network must be resilient, adaptable, and mobile.
    - **Resilient:** Uploaded files must still be recoverable even if several nodes in the mesh network are taken offline.
    - **Adaptable:** The mesh network should continue to function and adapt its data routing when nodes are introduced or removed.
    - **Mobile:** Nodes should be able to move freely, using battery power when needed, while still remaining routable to the rest of the mesh network. The mesh network should also be multi-hop, allowing nodes to communicate with other nodes not directly in range.

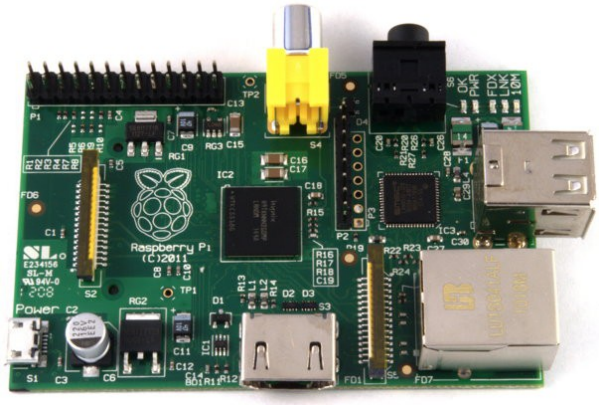
The technical inspiration was taken from the PirateBox project, a “self-contained mobile communication and file sharing device.”[1] Using a wireless router, the PirateBox creates a standalone wifi Access Point, not connected to the internet, allowing clients within range to share files and chat with each other. While a useful model for what I have in mind, the PirateBox's centralized architecture becomes a critical vulnerability. If a PirateBox were to be confiscated by authorities or somehow taken offline, all services and files on that PirateBox are no longer available. However, distributing the PirateBox's file-sharing server across a mesh network of nodes would distribute the vulnerability of any one node being taken offline. This distributed architecture, then, is how the MediaGrid operates.

Hardware

Other wireless mesh network projects often use wireless routers as their primary devices of choice. Consumer-grade wireless routers are generally cheap, relatively small, and of course have the required components needed for wireless networking.

1 http://wiki.daviddarts.com/PirateBox

However, there are two primary limitations in using wireless routers for the MediaGrid project. First, routers have limited processing power. While these devices are perfectly sufficient for general routing and even for small embedded servers, their CPU speed and RAM can be insufficient for more complex applications. Second, wireless routers generally have a very small amount of flash storage, often without means for attaching external storage, for instance via USB. Applications requiring large code libraries or an interpreter will not be able to fit on the router itself.



The possibility of running a mobile wifi mesh network with complex, distributed applications was significantly boosted with the recent release of the Raspberry Pi. The Raspberry Pi (Rpi) is a single-board bare-bones computer the size of a credit card. The latest Raspberry Pi model includes a 700 MHz processor, 256 Mb RAM, and two USB ports and an SD Card slot allowing for external storage. In addition, the RPi costs a mere $35 USD. The capability-to-price ratio of the RPi makes it ideal for low-cost, multi-node projects.

Although lacking embedded wireless capabilities, adding a USB wireless card brings the price to $55 USD, comparable to many low-end wireless routers, yet with more powerful hardware. For the wireless hardware, I chose the Alfa AWUS036NEH, which includes a 802.11b/g/n wifi radio that supports ad-hoc mode with drivers included in the linux kernel. The Alfa card can also use an external antenna with a RP-SMA connector. Finally, the Raspberry Pi can be powered with a battery source providing at least 700mA at 5V, connected to a microUSB port.

The Raspberry Pi runs several linux distributions compiled for ARM11 architecture, allowing for a wide variety of software. For this project, I have used Debian Wheezy, with future plans to create a custom Debian image, pre-configured with the MediaGrid software components.

ARCHITECTURE

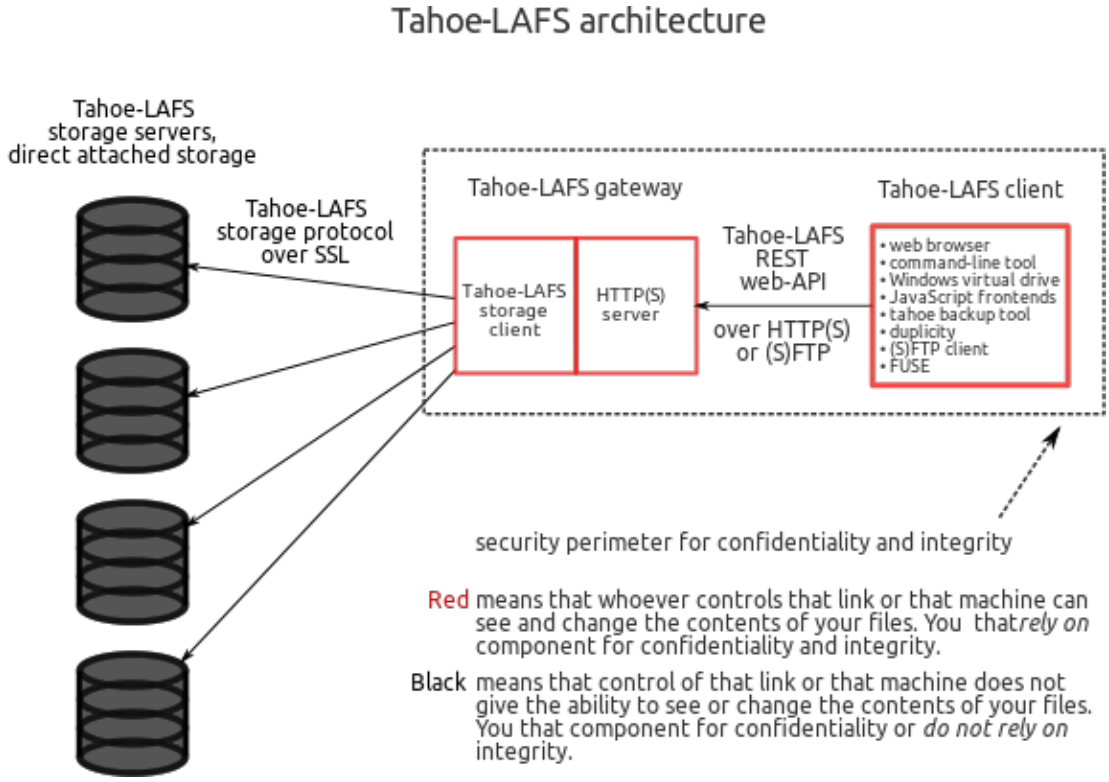
There are two main components to the MediaGrid project. The first deals with meshing the nodes, and the second involves the distributed filesystem. For a meshing protocol, I have chosen the widely-used Optimized Link State Routing Protocol (OLSR). The open source implementation of OLSR, called OLSRd[2], is available in the main Debian Wheezy repository, allowing for easy installation on the RPi. OLSR allows multi-hop mesh routing between nodes in an ad-hoc wireless network, and forms the backbone of the MediaGrid project. Each device in the MediaGrid will be running OLSRd, making it accessible via wifi to all the other devices.

olsrd
an adhoc wireless mesh routing daemon

2 http://www.olsr.org/

Tahoe-LAFS[3] serves the second component of MediaGrid: the distributed filesystem. Using either USB or SD Card storage, Tahoe-LAFS stores uploaded files in encrypted segments distributed across a number of nodes, such that the data will survive the loss of most of the nodes. Only a minimum number of nodes still need to be available in order for all the files to be recoverable. This number depends on the configuration of the MediaGrid.

The MediaGrid is constructed such that each device in the mesh network includes storage space for the Tahoe-LAFS filesystem. To make the MediaGrid as accessible as possible, each device will be running a simple Python web server that acts as a custom front-end gateway to the Tahoe-LAFS storage grid. Thus, a smartphone's web browser is all that is needed to upload files onto the MediaGrid. The web server can optionally use HTTPS to ensure encrypted communications. Uploaded files are also encrypted before being sent across the air to the other nodes in the grid. Using the Tahoe-LAFS diagram below[4], we can see that each device in the MediaGrid acts as both a Tahoe-LAFS gateway as well as a storage server:



Limitations and Vulnerabilities

- Communications are not anonymous, even when operating in darknet mode (not connected to the internet). This is because the 802.11 frame contains the unique MAC address of the communicating device, allowing for identification. An adversary passively monitoring wifi traffic could thus identify which clients are using the grid, although not necessarily what files they are sharing.
- Uploaded media files can further reduce the anonymity of clients due to the presence of meta-data, such as the device used to capture the photo/video, their GPS location, and the time the photo/video was captured.

3 https://tahoe-lafs.org/trac/tahoe-lafs
4 https://tahoe-lafs.org/~zooko/LAFS.svg

- An open, public MediaGrid is susceptible to a spoofing attack. An adversary can setup a rogue node posing as a node in the mesh network, thus capturing uploaded files intended to be stored on the MediaGrid, and identifying clients. This threat could be mitigated by pre-configuring clients to make connections only to authenticated grid nodes, perhaps through the distribution of a smartphone app.
- Since the mesh grid is accessible to anyone, the design currently does not prevent an attacker from uploading excessive amounts of data, consuming the limited bandwidth and storage space of the grid. This could possibly be mitigated by limiting access to the network to predetermined, authenticated users.
- Ad-hoc wireless mode has limited support among android phones and older hardware. Such devices would not be able to connect to the mesh network without an Access Point bridge.

Future Plans

- Incorporate a communications platform, allowing secure public and private webchat, as well as SMS integration. Possibly integrate with the March Hare Communication Collective's custom Ushahidi build.
- Allow for uploading private files, only viewable by the uploader or mesh creators. This should be easy to implement, as the capabilities are already included in Tahoe-LAFS.
- Automatically remove meta-data from uploaded media files.
- Develop an internet-connected feed-type publishing platform for uploaded media.
- Make a snazzy weather-proof case for the Raspberry Pi.