Baculus

A resiliant, self-contained, friendly internet buoy

Baculus is a clear plastic backpack with wheels, a telescoping antennae/flag, a wifi access point, small computer, gps transceiver, software defined radio, and battery. The platform provides ad-hoc, self organizing wifi networking, by working together with the humans who use it. By default, two mobile-first web and sms based applications run on baculus, to facilitate communication - a message board, and map checkin tool. We will share baculus with community leaders well before any possible disaster scenario. This allows for members to become comfortable adapting the sytem to their needs, so when disaster strikes, it is a natural response to use the network.

One key differentiator is what happens when existing mesh links go down. For this, we are using a self-repairing process where the boxes contain a phsyical wifi-divining rod that asks humans for help, by showing them which direction to bring the device to connect to the rest of the network.

baculus is

self contained - a literally transparent, waterproof backpack with room for people to modify to meet their own needs. All tools work fully on their own, with or without a larger network.

resiliant - the network asks humans for help repairing if a link goes down, by finding and directing where they should be placed.

friendly - the software design is minimal, allowing for communities to use in whatever way is most natural for themselves.

a buoy? - meant to be moveable by a single person when needed to fill gaps in network connectivity

baculus helps

communities define their own needs, with a simple message board, and editable map.

repair infrastructure during disaster, since disconnected nodes discover connected nodes and direct people where it should be setup

answer who is where when disaster strikes

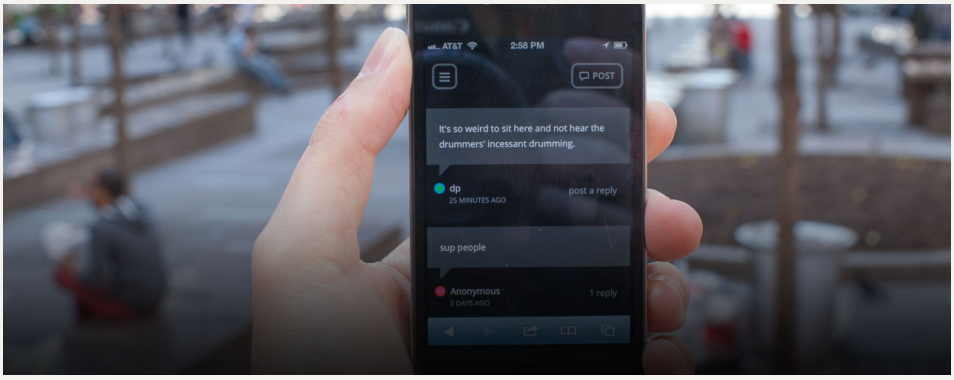
provide power and connectivity to help coordinate when needed

Components

Applications

There are two main interfaces for the initial software running on baculus - mobile web applications, and text messages.

We will be using the mobile interface from occupy here to allow people to quickly post to a global stream the most important information. It will be backed by simple text storage that will be synced peer-to-peer using the scuttlebutt protocol. This allows for intermittent connections, and no central server needed to coordinate. Special care will be taken to minimize both the initial application load and



Additional features built on top of the interface will be the gps location of the person who posted, and a toggle between map and stream view. The map view is especially targeted towards first responders, and will allow for location specific messages to be sent - for example, an SMS blast towards a particular base station to ask for a supply that is known to be there.

A slightly specialized version that is just a name/location registry will be built to quickly answer the question 'who is where', which is more crucial during the first 72 hours after disaster strikes.

For cellular phones without a web browser, openBTS will be used to create a cellular network where people can send and lookup stream messages via SMS. We have prototyped and successfully deployed this network for both an art show and a previous proposal with the Red Cross. We will be running an instance of the dumbstore to simplify the process of creating new applications on top of osmoBTS.

```
class Weather < Dumbstore::App
    author 'Jonathan Dahan'
    description 'Sample weather app'

    text_id 'weather'
    def text(params)
      current_weather = WeatherApi.get_weather
      "<Response><Sms>The weather is #{current_weather}
</Sms></Response>"
    end
end
```

> *example dumbstore application, also supports responding via voice*

Operating System

We will be running the most upstream version of raspbian. Any modifications of the base image will be shared as reproducible build scripts built on top of the PiBakery framework. Any custom modules will be upstreamed to the PiBakery project. We already have experience upstreaming libraries to this framework.

The cellular network operating system will be built on top of OsmoBTS, which already has support for the LimeSDR.

Hardware

The system is designed to work with off-the-shelf hardware, to show the true cost for any participants to repair or replace components as needed.

Raspberry Pi 3 - $45 (with microSD, usb) ubiquitious, open platform with great community support and availability.

Ubiquiti Litebeam AC - $55 good point to point wifi that spans kilometers. We will be mounting these on a rotateable antenna to provide for the widest possible coverage. What is nice

Plastic backpack w/wheels - $35 from the container store / amazon

1kWh battery - $85 should last 72 hours + be able to charge devices, which are the most crucial times for responding to needs.

RTL-SDR - $20 which can receive Iridium satellite messages for coordination

Total cost is $240 per node.

Optional

100W solar panel - $99 This may prove cheaper than electricity in some areas, and makes more sense when you think of this infrastructure as being present and useful in non-emergency as much as disaster response.

Lime SDR Mini - $180 full duplex SDR that can support 2g/3g/4g cellular networking, and iridium satellite messages

Networking

For WiFi networking, we will be using the QMP images, customized scripts will be published in a git repository. For the optional cellular SMS, we will use OsmoBTS, and for Iridium we will use gnu radio.

Deployment

Ket to Baculus being actually useful is its integration into communities before disaster strikes. To do that, the software needs to be simple, flexible, and useful. Previous projects such as Red Hook Wifi has shown that editable maps are intuitive and useful. Textual interfaces are well understood and flexble.

Repair Process

Physical network repair

A model to think about our physical / network repair process is the rain cycle. Nodes that have an uplink to the Iridium satellite network, either directly or through an internet gateway service precipitates its GPS coordinates up. Every few minutes, whatever coordinate data exists are blasted down to all the nodes using Iridium burst. These bursts, recievable **anywhere in the world**, allows disconnected nodes to find their closest neighbor. After being brought close enough to connect via wifi, that nodes gps location gets added to the cloud.

> *This might be too damn close to cloud computing, maybe theres a better analogy, or no analogy*

But how do we get the node where it needs to go? There are four places where a person can find out what direction, and how long it will take, to bring the node close enough to another. The four places that information can be found are:

1. The map when connected to wifi (requires device with screen)
2. The SMS gateway when connected to 2g (requires cell phone)
3. Compass + seven segment display (connected to pi)
4. Custom divining rod (optional, requires testing)

The custom divining rod is a prototype physical interface we would like to test out, to see if it is helpful and reliable. A digital compass in the rod will allow a person to point it in any direction, and have it vibrate and light up in the direction it should be carried. An estimated number of minutes will be displayed so they understand just how far it might be.

What happens if Iridium is down?

If Iridium is down, we fall back to using the RT-SDR to do frequency ping/scanning sweeps. Protocols[^such as?] have been used to broadcast 'hello I am here' signals. We would start at the lowest frequency gradually increasing if nothing is found. This will be another large part of the testing work.

What happens when wifi goes down?

If wifi signals are failing, we can experiment with data-over-fm for critical information (a name/location registry)

Information Repair

How does the software work when they join the network? We will be using scuttlebutt, a well known and testing protocol for sharing information with intermittent network connectivity. This combined with statebus to reconcile differences allows us to opportunisticly share information without the frontend application worrying about having to be online.

> *This is a collaboration between the humans and the machines, and I would love to have the software, hardware, writing, etc, have a tone showing that the machines need help. I'm not sure if during a disaster that is insensitive or distracting, but just sharing my thoughts.*

During a disaster

In the first 72 hours after a disaster, people immediately look for power, usually to charge their phones. They want to make sure they can communicate with loved ones, and coordinate help if they are in a position to do so.

Design Values

Off the shelf hardware to have maximum accessibility

The raspberry pi has tons of community documentation, and a good repair story

Existing protocols for maximum compatability

Using well known protocols like AM/FM radio and WiFi mean that people can build baculus-compatable systems without special expertise, hardware, or programming.

Platforms over Products

A product requires support to maintain, and is often harder to write import/export tools. Plain text databases, forums, are adaptable to many needs.

Upstream everything

Any extensions, plugins, build scripts, etc, will be brought into the upstream project. Having experience with contributing to many open source projects, package managers for macOS and multiple linux distributions means it is important to us that the work can be used and modified by as many people as possible.

Feasability

Technical

The most complicated component of baculus is communicating with the Iridium satellite network. This has been shown to work with an RT-SDR and HackRF in ccc talk link here. Independently, we have been able to tune into Iridium with our local RT-SDR but still need to work on making it robust and writing the software to send and decode GPS data. For running the SMS interface, we have already build openBTS cellular networks, and had successfully gotten over 40 cell phones to autoconnect, receive broadcasts from a locally running web application, and send SMSs back during arthackday link here.

We also have experience writing offline first open source web applications, from small prototypes that use no libraries or frameworks like this one-file schedule sketching application that can share state in the url to synchronized collaborative wiki editing link here.

Human

The most complicated non-technical component of baculus is getting it into the hands of community leaders. From our experience with NYC Mesh, having interested, reliable advocates are the only way for people to use the system and nodes to stay up during non-disaster times. We have had limited success working with The Red Cross in a previous grant link here , and are hoping that this challenge will help us connect more with people who need it most. Locally, during the prototype phase, we will be testing with leaders from the Red Hook Initiative, Pioneer Works, and others in Red Hook and Gowanus.

Deliverables

For the prototype phase, we plan on having at least the following:

A dozen nodes built

Two field tests, one in Red Hook, one in a location outside of the US TBD

Website with instructions how we build and deploy

Markdown documentation with methods/results of case studies and user tests

Wiki for others to fork/improve on the project

Gitlab + github repositories for any custom code that could not be upstreamed

Link Here

> *This is just a placeholder for links*