Woodchuck: Improving Data Availability for Mobile Devices

Neal H. Walfield

GHM 2011
August 26, 2011

Data Availability: n. The degree to which data that is needed or desired is accessible.

Data Availability: n. The degree to which data that is needed or desired is accessible.

> *“Thanks to Woodchuck, my data availability has increased dramatically!”*
>
> — *Woodchuck PR Team Leader*

Outline

- Problem
- Approach
  - Solution
  - Effectiveness?
- Status

What's the Problem?

You leave the house...

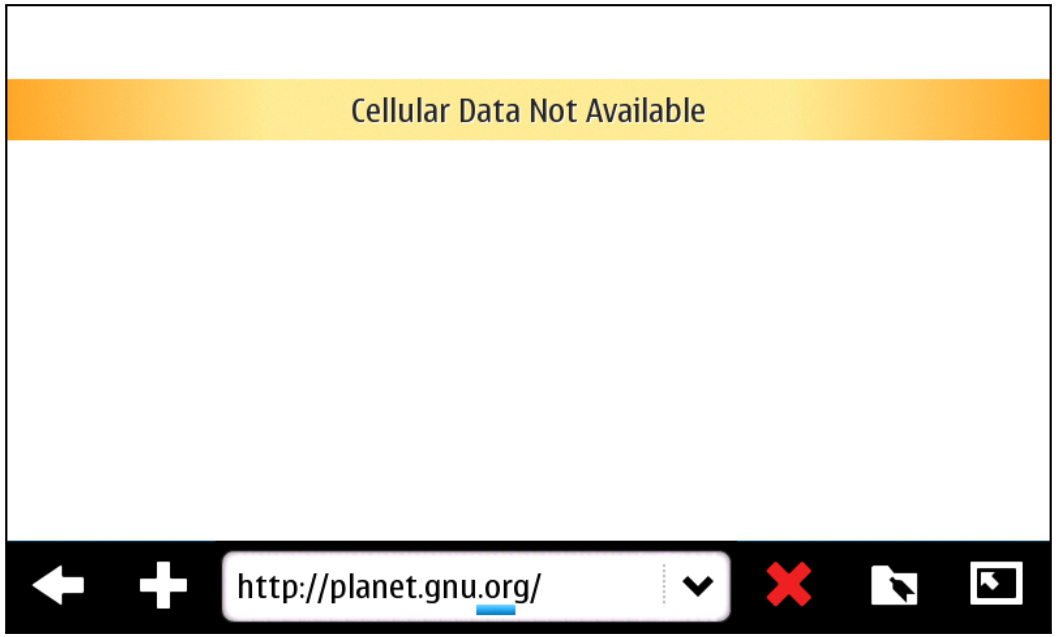
...get in the train...

...and turn to your mobile device for...

- Blogs, μ-blogs, social network updates
- Podcasts
- *Email*
- *Calendaring*

...and, you wait...

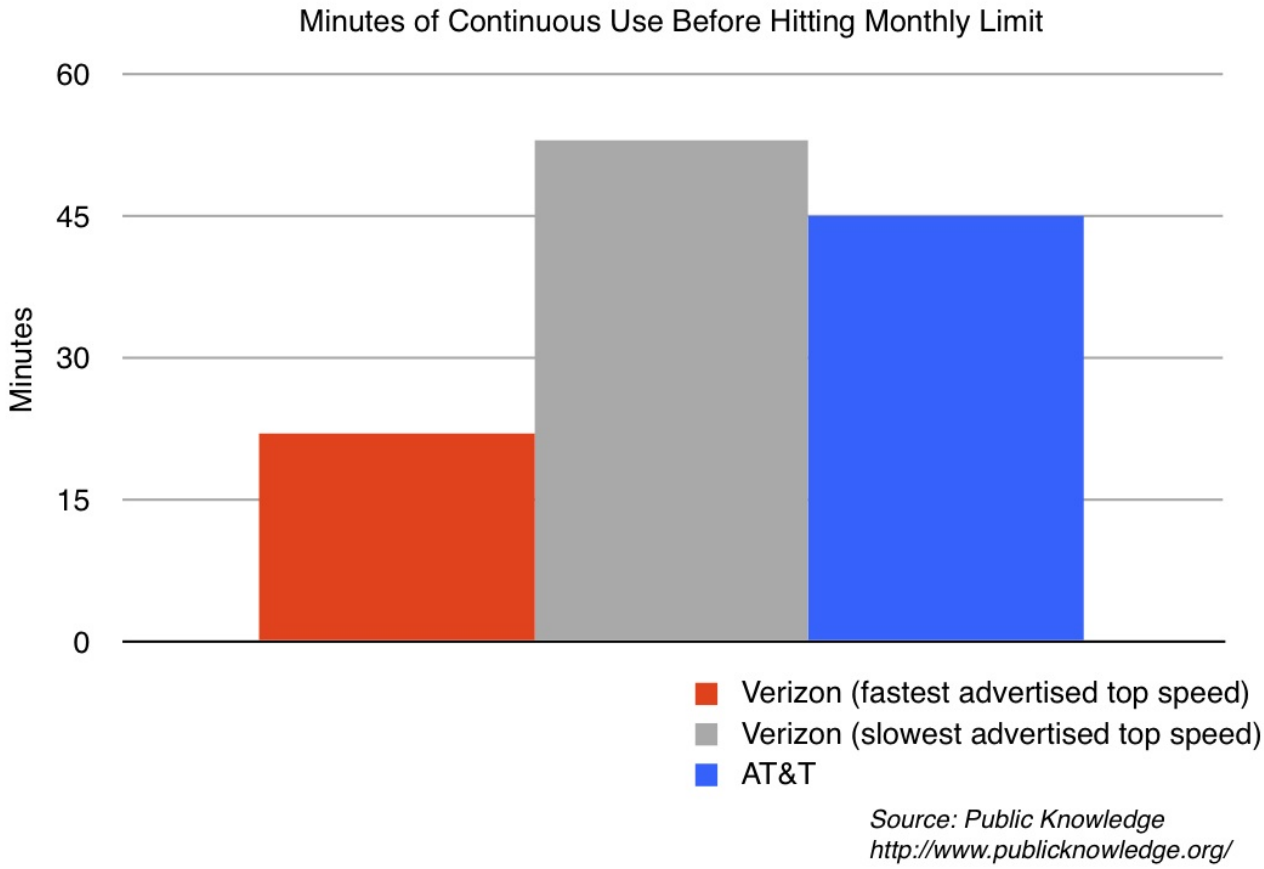
...but, connectivity is poor...



- How poor? Around Houston:[1]
  - Probability of connecting to a cell tower: 99%
  - Probability of creating a data connection: 80%

[1] Ahmad Rahmati and Lin Zhong, “Context-Based Network Estimation for Energy-Efficient Ubiquitous Wireless Connectivity,” 2011.

...data transfers are expensive...



From: Arbitrary Data Limits Make Wireless 4G A Waste of Money, Michael Weinberg, 2011. `http://www.publicknowledge.org/blog/arbitrary-data-limits-make-wireless-4g-waste-`

...and wireless drains the battery...

Access	Activity	Watts	Ratio
3G	Play 56.Kb/s stream	**1.00**	12.5
Edge	Play 56.Kb/s stream	0.96	12.0
WiFi	Play 56.Kb/s stream	**0.75**	9.3
Flash	Play 320.Kb/s files	**0.32**	4.0
—	Idle	**0.08**	1
—	Idle, LCD on	0.27	3.4

Energy used by a Nokia N900. Battery has 5 Wh.

Observations

- Much data is delay tolerant
  - Receiving
  - Sending
- User explicitly subscribes to “data streams”

Solution

- Prefetch downloads
- Queue uploads

System Structure

- Each application monitors connectivity?
  - $\Longrightarrow$ All applications run in background
  - $\Longrightarrow$ Duplicated effort
- How to coordinate use of:
  - data transfer budget?
  - energy?
  - storage?

Being Smart

- Hourly news on commute home?
  - Want news from 5pm, not 6am!
  - Only downloading with WiFi and power is insufficient!

Woodchuck

- Observe environment
- Observe user behavior
- Predict needed/desired data
- Predict connectivity
- Schedule transfer smartly

Observing the Environment

- Connected cellular towers
- Wifi access points
- Quality of service: 10 Mb/s or 10 kb/s?

Observing the Environment

- Connected cellular towers
- Wifi access points
- Quality of service: 10 Mb/s or 10 kb/s?

- Privacy: **Hash data with a private salt**

Observing User Behavior

- What data is used?
- Where? When?
- How?
  - Sequential, e.g., TV Series
  - Only newest, e.g., News

Observing User Behavior

- What data is used?
- Where? When?
- How?
  - Sequential, e.g., TV Series
  - Only newest, e.g., News
- $\Longrightarrow$ Application support
  - Register streams/objects
  - Publication time, download time
  - Object use

Predicting

- Locations in the near future
  - Graph of cell tower transitions
- Needed data
  - What streams have been used in predict locations?
  - How? Object publication time to use?
- Compute data/power budget
  - Now
  - At each location

Transferring

- Woodchuck makes upcalls to application
  - Update stream
  - Transfer object with quality X

Murmeltier

- Woodchuck implementation
  - Packages for Maemo 5, Debian
- DBus interface
- glib-based C library
- Python module

[2]

[2] By romkey, CC BY-NC 2.0

Application Changes

- Register streams
- Listen for Woodchuck upcalls
- Notify Woodchuck server of events

Registering Streams

```
stream_ids = [s.identifier for s in wc.streams_list()]

# Register any unknown streams.
for key in self.getListOfFeeds():
    title = self.getFeedTitle(key)
    if key not in stream_ids:
        # Use a default refresh interval of 6 hours.
        wc.stream_register(key, title , 6 * 60 * 60)
    else:
        # Make sure the human readable name is up to date.
        if wc[key].human_readable_name != title:
            wc[key].human_readable_name = title
        stream_ids.remove(key)

# Unregister any streams that are no longer subscribed to.
for id in stream_ids:
    wc.stream_unregister(id)
```

Handling Upcalls

```
class woodchuck (PyWoodchuck):
    def __init__(self, feeds, human_readable_name, dbus_name):
        PyWoodchuck.__init__ (self, human_readable_name,
                              dbus_name)
        self.feeds = feeds

    def stream_update_cb(self, stream):
        self.feeds.updateFeed(stream.identifier)

    def object_transfer_cb(self, stream, object,
                           version, filename, quality):
        pass
...
for article in articles:
    wc[feed].object_transferred(
        object_size=article.size,
        publication_time=article.publication_time)
wc[feed].updated(new_objects=len(articles))
```

Notifying Woodchuck of Events

```
wc[feed][article].used()
```

Evaluation

- What algorithms are effective?
- User study:
  - Anonymized location
  - Connectivity
  - Files accessed
  - Programs used

Ported software

- FeedingIt, an RSS Reader: N900 packages available
- gPodder, podcast manager: patches sent upstream
- Khweeteur, identi.ca, twitter client: almost done

Summary

- Goal: Improve data availability
  - Hide spotty network coverage
  - Manage data caps
  - Use energy more efficiently
- Solution:
  - Exploit delay tolerant data
  - Predict what is likely needed

`http://hssl.cs.jhu.edu/~neal/woodchuck`

N900 Packages:

`http://hssl.cs.jhu.edu/~neal/woodchuck/woodchuck.install`