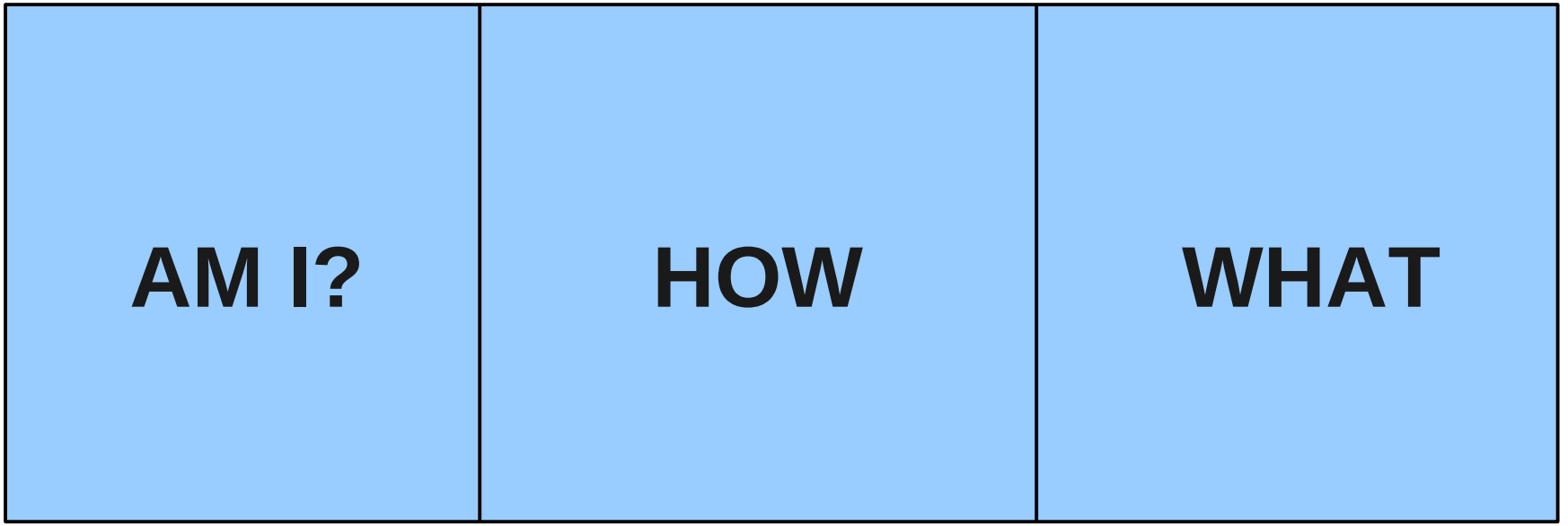
Network and Location Awareness in Your Application

Dan Williams

redhat

Desktop summit

There are three core parts of Network Awareness...



Am I connected at all?

The How: what is the “cost”of a packet?

$ € ¥

redhat

Consider security...

And latency...

The What: what am I connected to?

EVIL?

redhat

Desktop summit

The What: what am I connected to?

Good...

Zones can help determine what apps should do using general categories.

Home

Work

Public

The connection tells you how to access the resources your app will use.

Location awareness is knowing where you are on a map.

Your mom is here...

redhat

Desktop summit

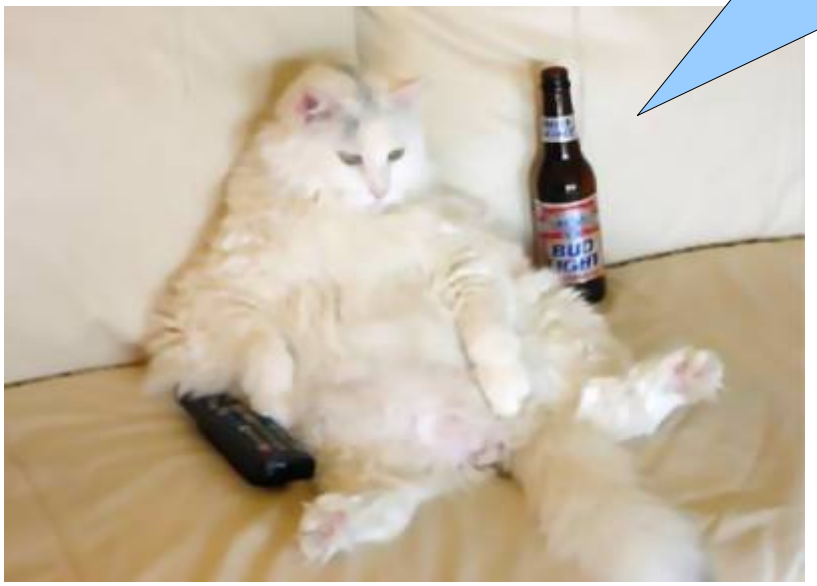
Each method of location positioning has different constraints and varied accuracy.

Slow but accurate

Fast but inaccurate

redhat

Desktop summit



redhat

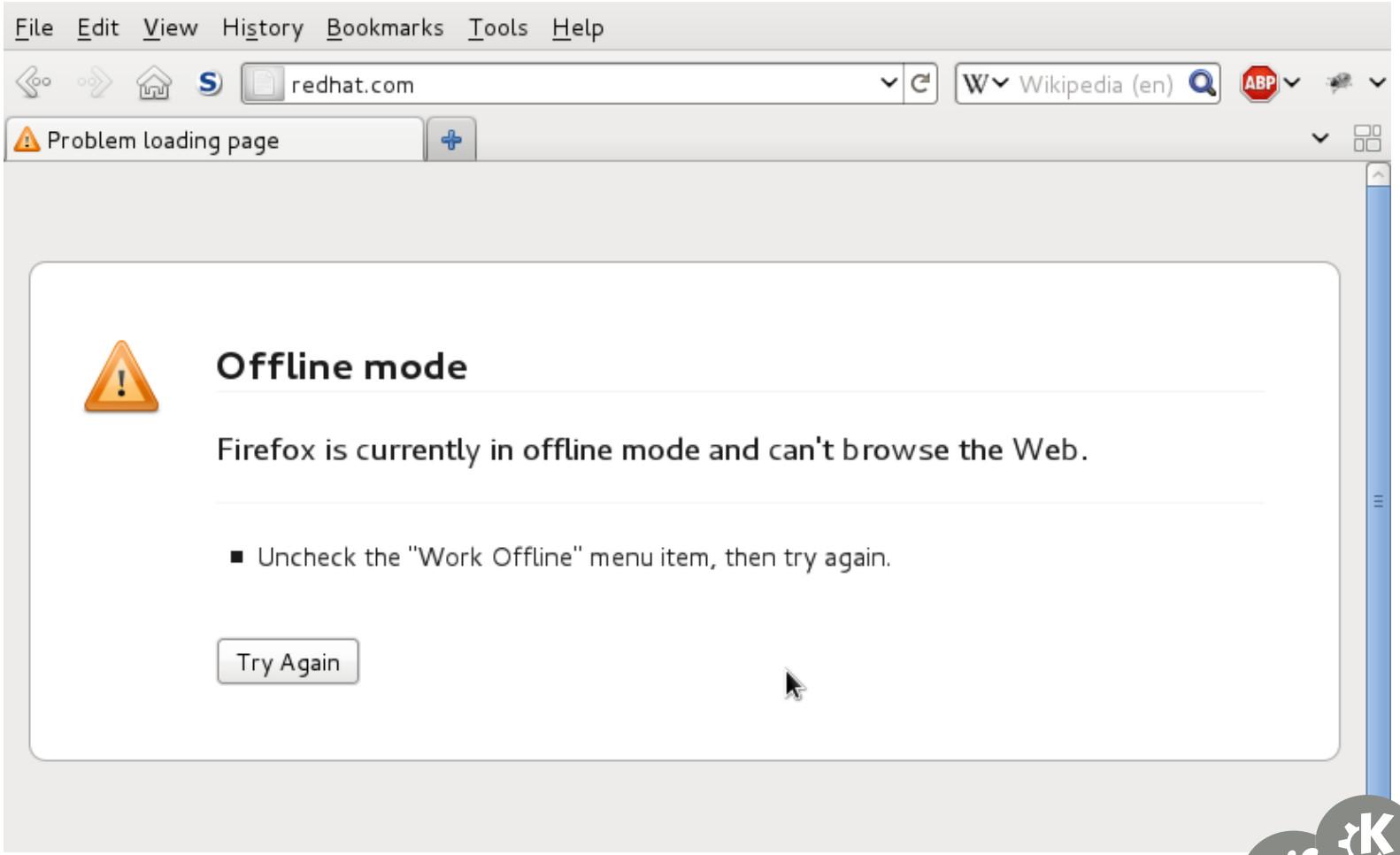
Desktop summit

You care about network and location awareness because you care about your users.

redhat

Desktop summit

It's trivial for many applications to do the smart thing.



redhat

Desktop summit

Location is the hot new thing in social apps.

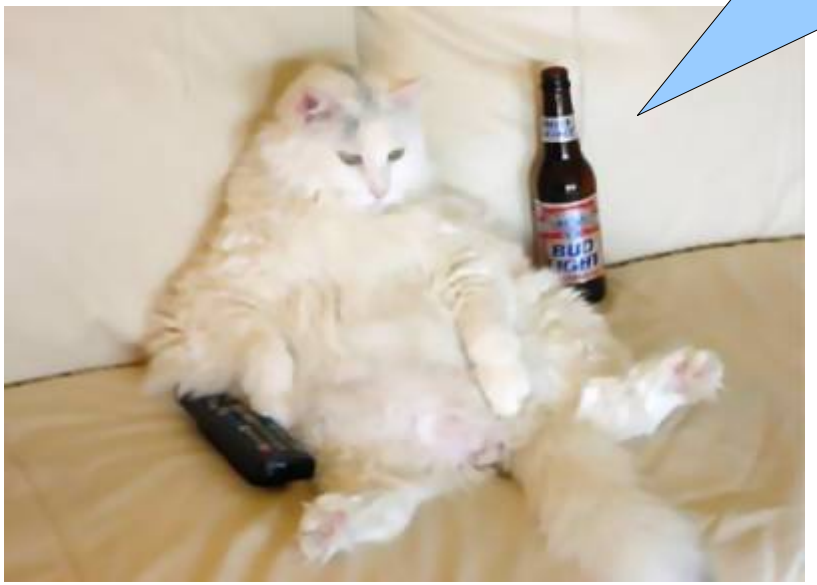

WHOA...!

Also keep the downside of location awareness in mind.

redhat

Desktop summit

That's all interesting, but what can I do about it?



redhat

Desktop summit

NetworkManager give your app the network awareness it so desperately wants.

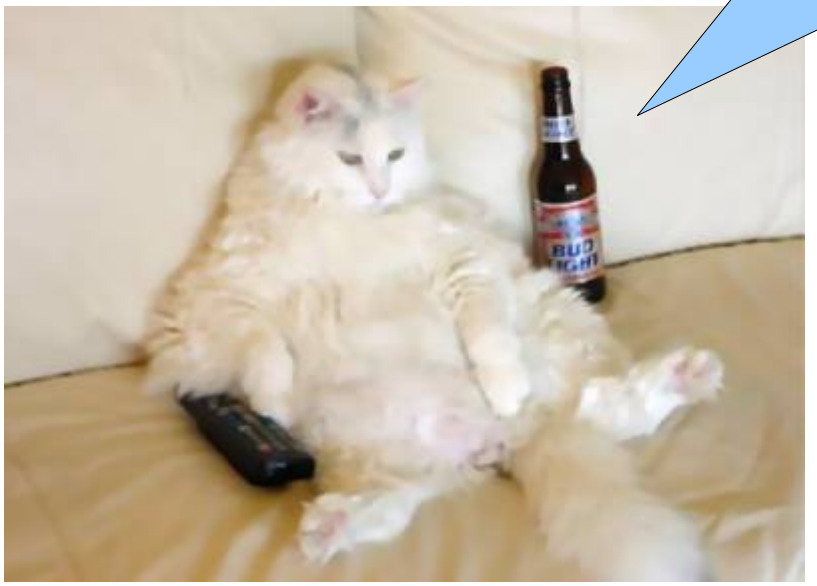
Decision	Status	Configuration

ModemManager prevents you from cutting your eyes out with a dull spoon.

redhat

Desktop summit

Most of the time you won't use ModemManager directly, but through a higher level framework.



redhat

Desktop
summit

Getting network state with NetworkManager is pretty trivial.

```
import dbus, sys
bus = dbus.SystemBus()

m_proxy = bus.get_object("org.freedesktop.NetworkManager",
                         "/org/freedesktop/NetworkManager")
manager = dbus.Interface(m_proxy, "org.freedesktop.NetworkManager")

names = { 0: "unknown", 10: "sleeping", 20: "disconnected",
    30: "disconnecting", 40: "connecting", 50: "connected locally",
    60: "connected sitewide", 70: "connected globally" }

state = manager.state()
try:
    print "State: %s" % names[state]
except KeyError:
    print "State: unknown"
```

Also trivial using KDE's Solid framework, if that's your thing...

```
If (Solid::Networking::status() == Solid::Networking::Connected)
{
    kDebug() << "Networking is enabled. Feel free to go online!";
}
else
{
    kDebug() << "Network not available.";
}
```

Getting the list of active network connections is pretty easy too.

```
import dbus, sys
bus = dbus.SystemBus()

# Get a proxy for the base NetworkManager object
m_proxy = bus.get_object("org.freedesktop.NetworkManager",
                         "/org/freedesktop/NetworkManager")
manager = dbus.Interface(m_proxy, "org.freedesktop.NetworkManager")
mgr_props = dbus.Interface(m_proxy, "org.freedesktop.DBus.Properties")

active = mgr_props.Get("org.freedesktop.NetworkManager", "ActiveConnections")
for a in active:
    a_proxy = bus.get_object("org.freedesktop.NetworkManager", a)
    a_props = dbus.Interface(a_proxy, "org.freedesktop.DBus.Properties")
    uuid = a_props.Get("org.freedesktop.NetworkManager.Connection.Active", "Uuid")
    print "%s" % uuid

if len(active) == 0:
    print "No active connections"
```

I like Qt, how would I list all saved network connections using it?

```
#include "NetworkManager.h"

void listConnections(QDBusInterface& interface) {
    QDBusReply<QList<QDBusObjectPath> > result = interface.call("ListConnections");
    foreach (const QDBusObjectPath& connection, result.value()) {
        qDebug() << connection.path();
    }
}

int main() {
    QDBusInterface interface(
        NM_DBUS_SERVICE,
        NM_DBUS_PATH_SETTINGS,
        NM_DBUS_IFACE_SETTINGS,
        QDBusConnection::systemBus());

    listConnections(interface);
}
```

How about a hot cellular positioning example with geoclue?

```
GeocluePosition *pos;
GeocluePositionFields fields;
double lat, lon;

pos = geoclue_position_new ("org.freedesktop.Geoclue.Providers.Gsmloc",
                            "/org/freedesktop/Geoclue/Providers/Gsmloc");
fields = geoclue_position_get_position (pos, NULL,
                                        &lat, &lon, NULL,
                                        NULL, &error);
if (error) {
    g_printerr ("Error getting position: %s.\n", error->message);
    goto done;
}

if (fields & GEOCLUE_POSITION_FIELDS_LATITUDE &&
    fields & GEOCLUE_POSITION_FIELDS_LONGITUDE) {
    g_print ("We're at %.3f, %.3f.\n", lat, lon);
} else {
    g_print ("Gsmloc has no location information.\n");
}
```

Questions?

redhat.