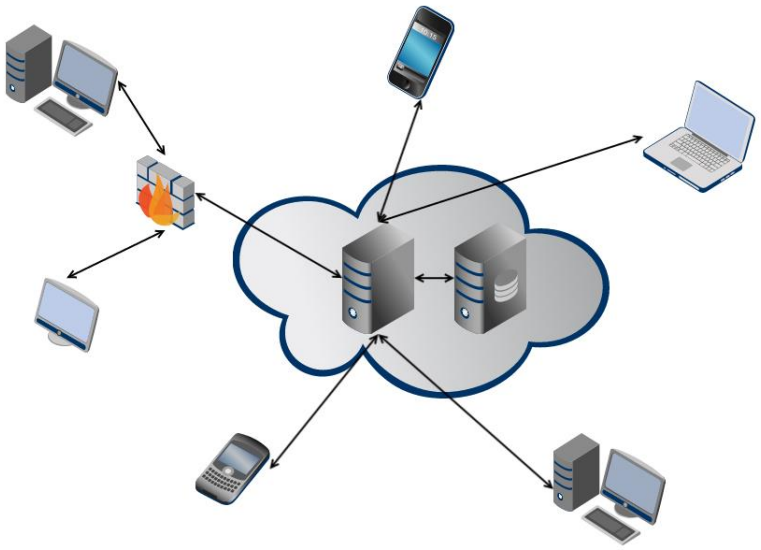
DIAS: Decentralized Internet Applications and Services

Final Project Presentation
4/22/14

Adriana Flores
Clayton Shepard
Ellis Giles
Yanda Lu
Haiuhua Shen

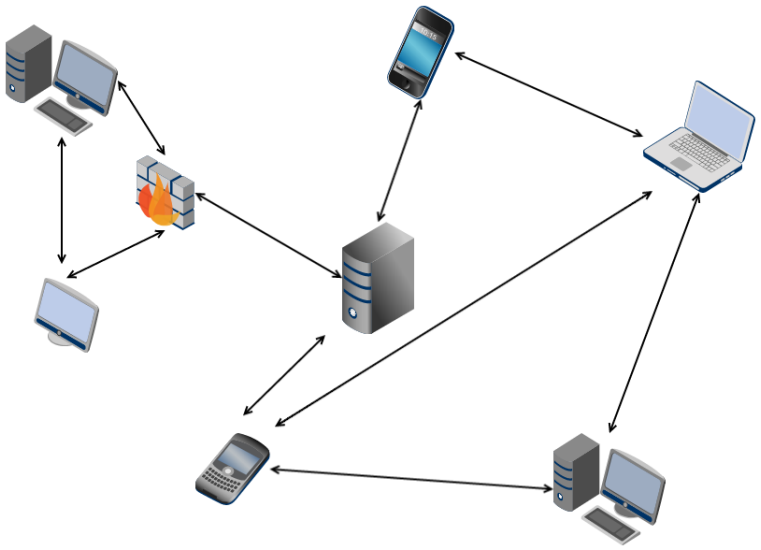
Problem: Centralized Architecture



- Single point of failure
- Privacy
- Security
- Control over data
- Energy efficiency
- Power hungry data centers
  - Google: 260 million watts [1]

[1] Glanz, James . "*Google Details, and Defends, Its Use of Electricity.*" *The New York Times 8 Sep. 2011*

Decentralized Architecture



- ✓ Robust to failure
  - ~~Single point of failure~~
- ✓ Full ownership and storage of your data
  - ~~Privacy~~
  - ~~Security~~
  - ~~Control over data~~
- ✓ Self-devices energy consumption
  - ~~Energy efficiency~~
  - ~~Power hungry data centers~~
    - ~~Google: 260 million watts [1]~~

DIAS: Decentralized Internet Applications and Services

Goal:

- Decentralize the current server-client model
- Replace servers with point to point communication for personal communication and services

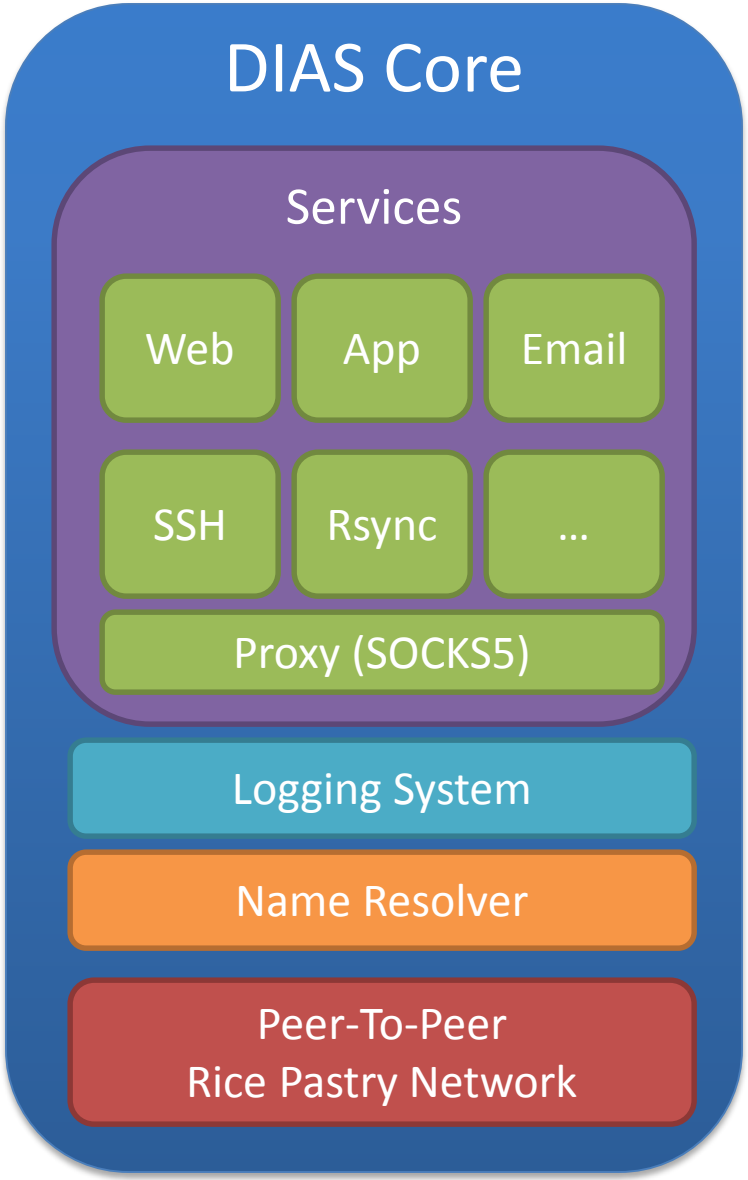
Benefits:

- Security
- Privacy
- Resilience
- Cost
- Power

Challenges:

- Redundancy
- Uptime
- Failover
- Battery Management

DIAS Architecture



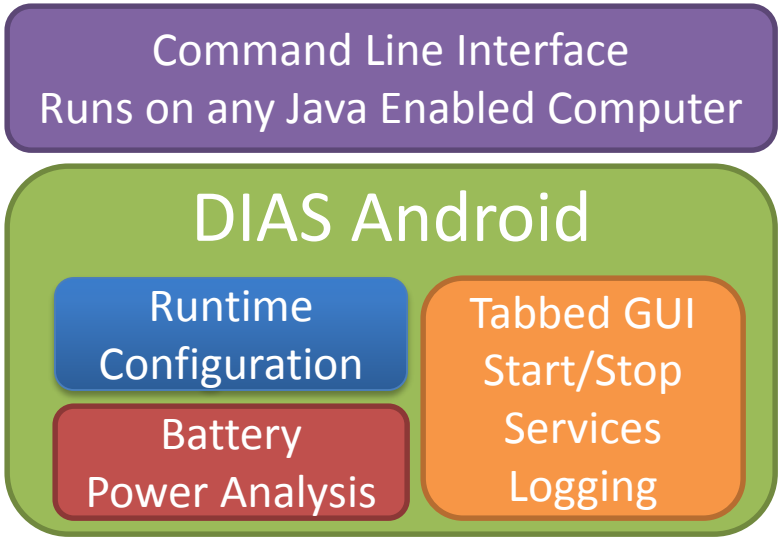
Different Services

Problem:

- Roll their own logging
- Name resolution
- Configuration Files
- Saving / Reading Data Files
- Different User Interfaces

Solution:

- Common Service Interface for Utilization!

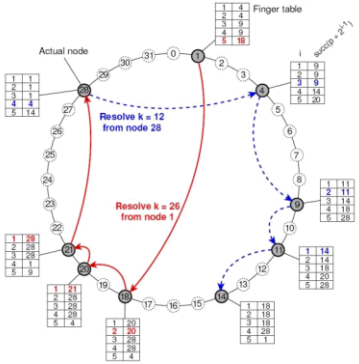


DIAS Implemented Services

Web

Proxy

SSH

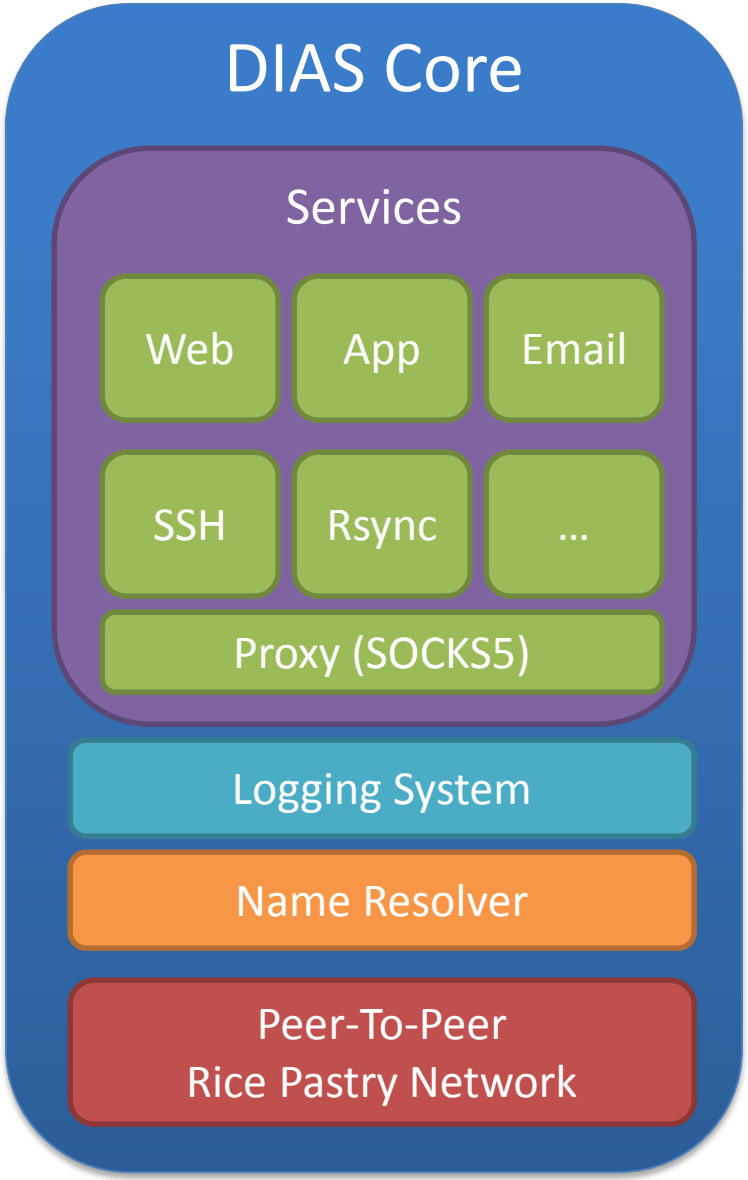


Peer to Peer

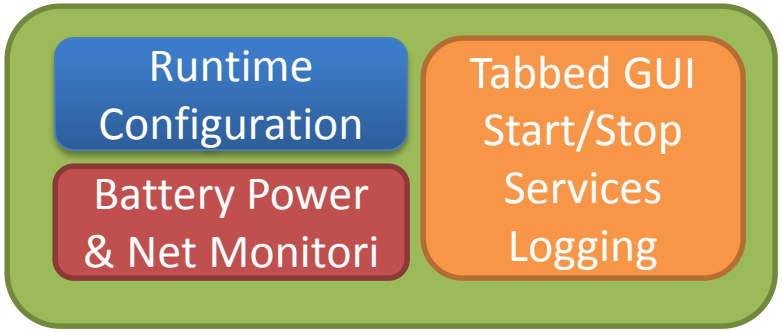
Email

Replication

DIAS Architecture: Threads



Function	Threads
Web Server	5 Threads
Application Server	5 Threads
Email Server	2 Threads
SSH Server	2 Threads
Rsync	1 Thread
Proxy Server	5 Threads
Name Resolver	1 Thread
Rice Pastry	2 Threads
Monitoring	1 Thread
Control Threads + Async. User & System Events	

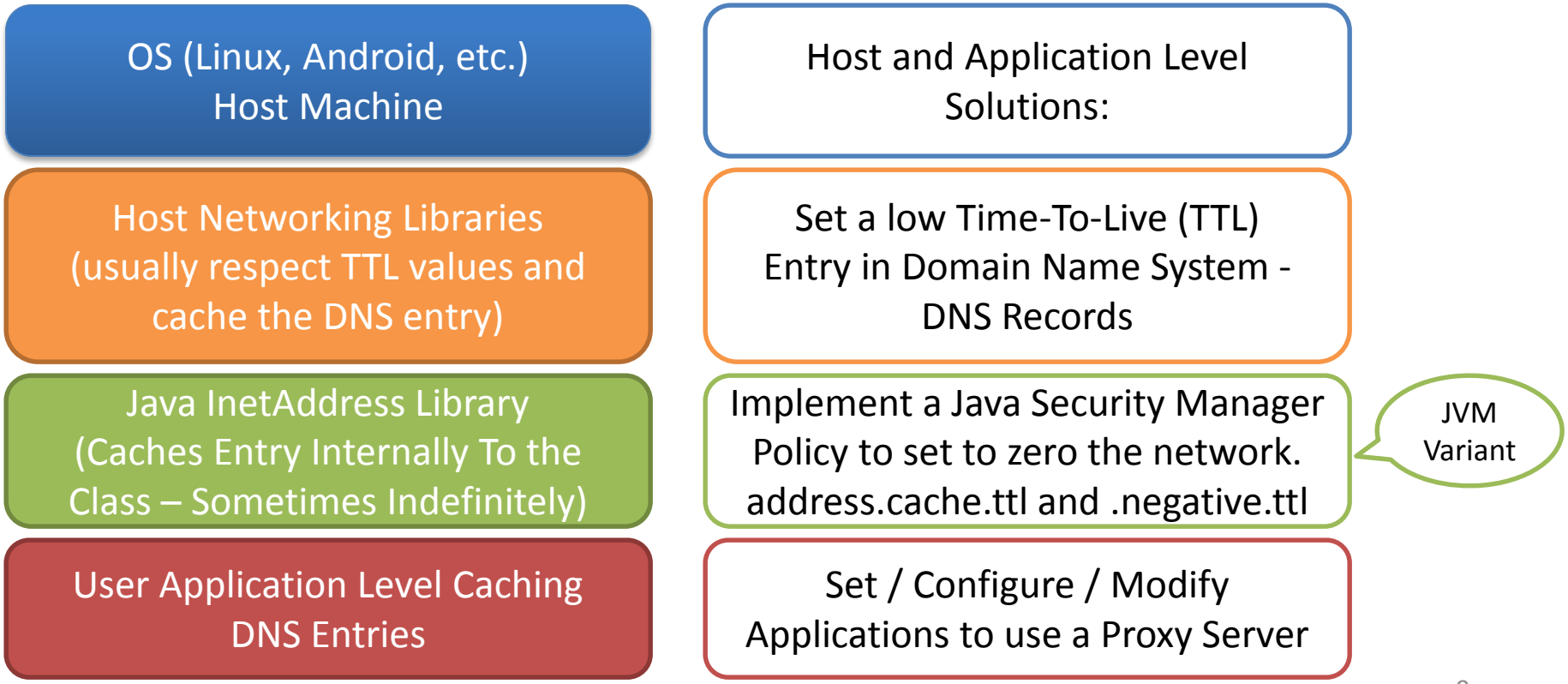


DIAS Architecture: Node Naming

- Suppose I want to run DIAS on my machine
  - How can others find my node if I need a fully qualified domain name?
  - Virtual Private Network can provide a central naming authority but becomes a central point of failure – against a decentralized Internet
  - Dynamic DNS solves some issues such as host IP changing, but hosts can be NAT'd behind firewalls or not available with stale entries.
  - One solution is provide naming resolution through an overlay network.
- Browsers and Applications will cache my node name too, so what happens if I change IP or failover?
  - DIAS users (or others) can use a service that will lookup names through the overlay network or DNS and not cache names.

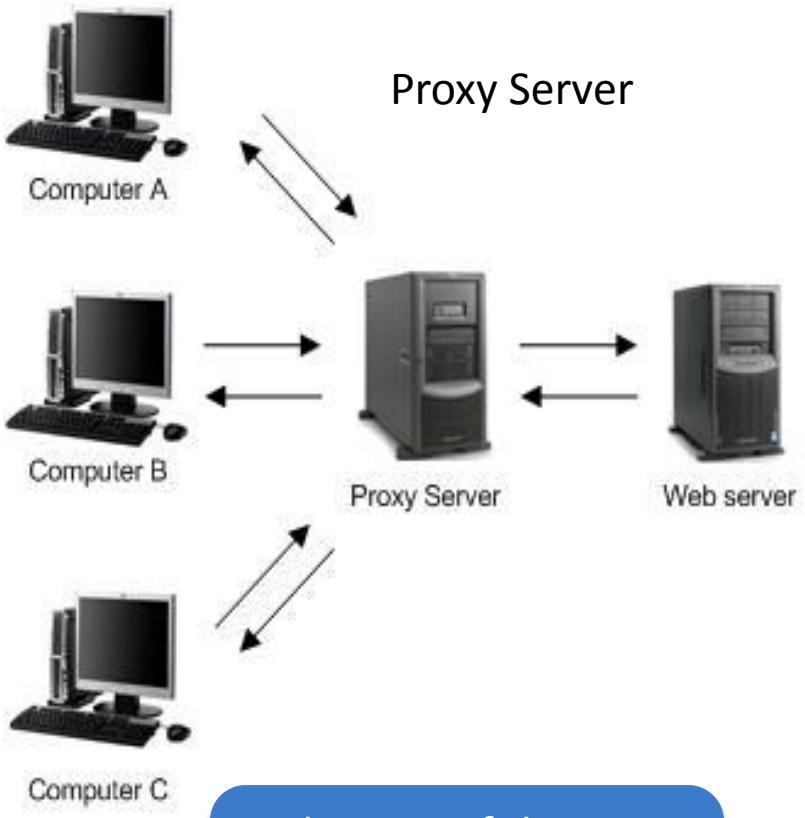
DIAS Architecture: Node Naming

- Domain Name System – DNS Entry Time-To-Live
- Host Caching of Domain Names

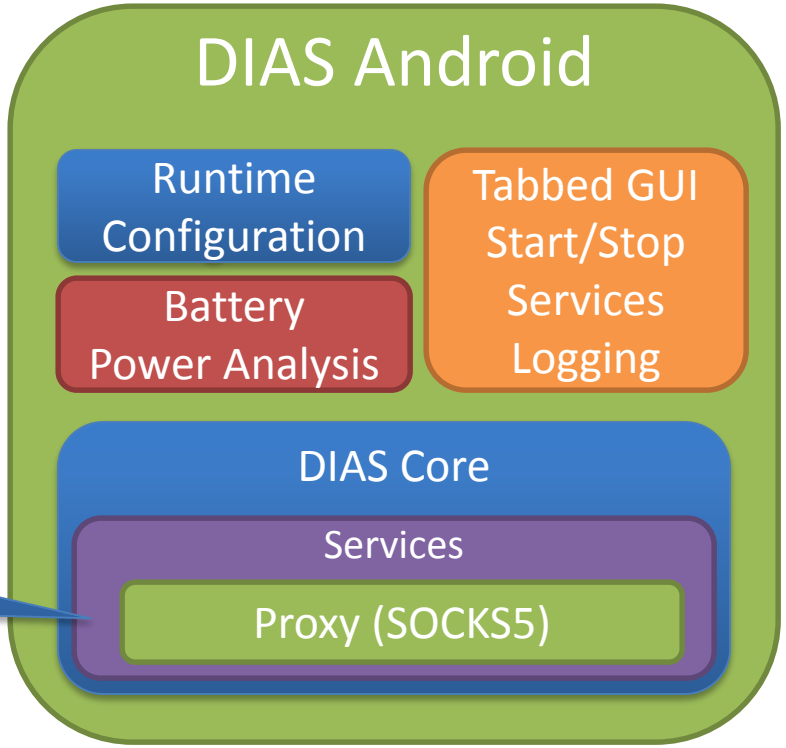


DIAS: PROXY

DIAS Architecture: Proxy



Now Embed the Proxy Server as software with DIAS…



Utilization of the proxy on localhost server can be problematic.

DIAS Architecture: Proxy

- Specifying the Proxy Server to localhost on a Wi-Fi Connection on Android doesn't work.
  - There's no bypass rules, so all traffic goes through the localhost proxy.
  - Thus, when the localhost proxy tries to reach the network, it is proxied itself and eventually crashes.
- Specifying a Proxy Server on a cellular connection isn't recognized.
- Google Chrome and Android's built in Internet browser don't support user proxy server.
- Use Mozilla Firefox or embed a Web browser (Web View and set proxy by Java Reflection into core class provided by Android).

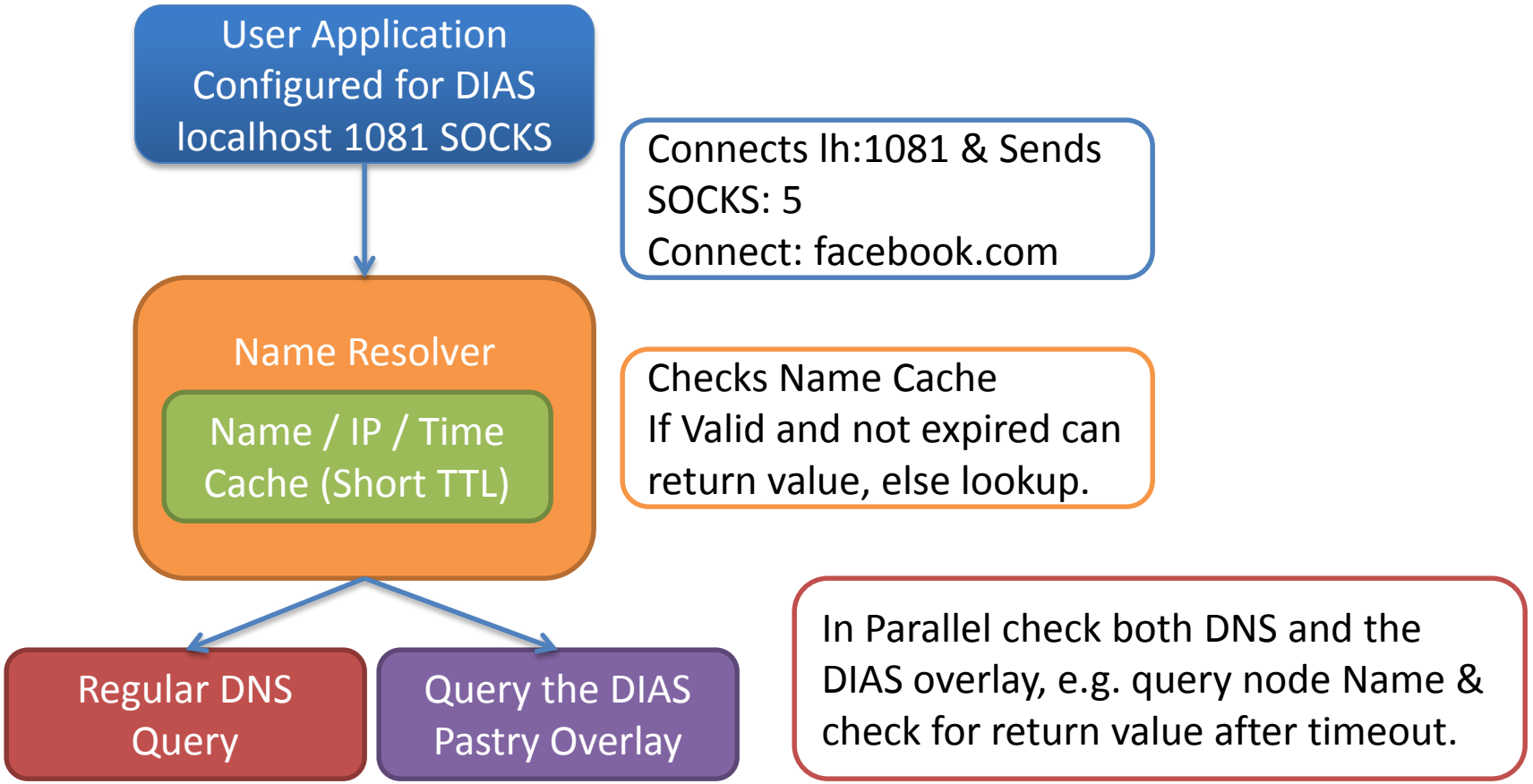
DIAS Architecture: Proxy

- Mozilla Firefox has Socks Proxy support built in for both desktops and Android.
- Set parameters to use Socks 5.
- about:config
  - Search for proxy:

network.proxy.socks	user set	string	localhost
network.proxy.socks_port	user set	integer	1081
network.proxy.socks_remote_dns	user set	boolean	true
network.proxy.socks_version	default	integer	5
network.proxy.ssl	default	string	
network.proxy.ssl_port	default	integer	0
network.proxy.type	user set	integer	1

Domain/Node Name Resolution

- Now we have a proxy, so what?



```java
package rice.comp529.dias;

import java.util.Vector;

public interface ServicesManager {
    public ServicesManager getServicesManager();
    public void registerService(ServicesInterface service);
    public Vector<ServicesInterface> listServices();
    public void logMessage(String message);
    public void logEvent(String event);
}
```

Proxy Jsocks
1081

Nano HTTPD
Web
8085

```java
package rice.comp529.dias;

public interface ServicesInterface {
    public String getDescription();
    public String getConfigUrl();
    public void start();
    public void stop();
}
```

Other Services

RSync

App
8080

Email
2525

SSH
22

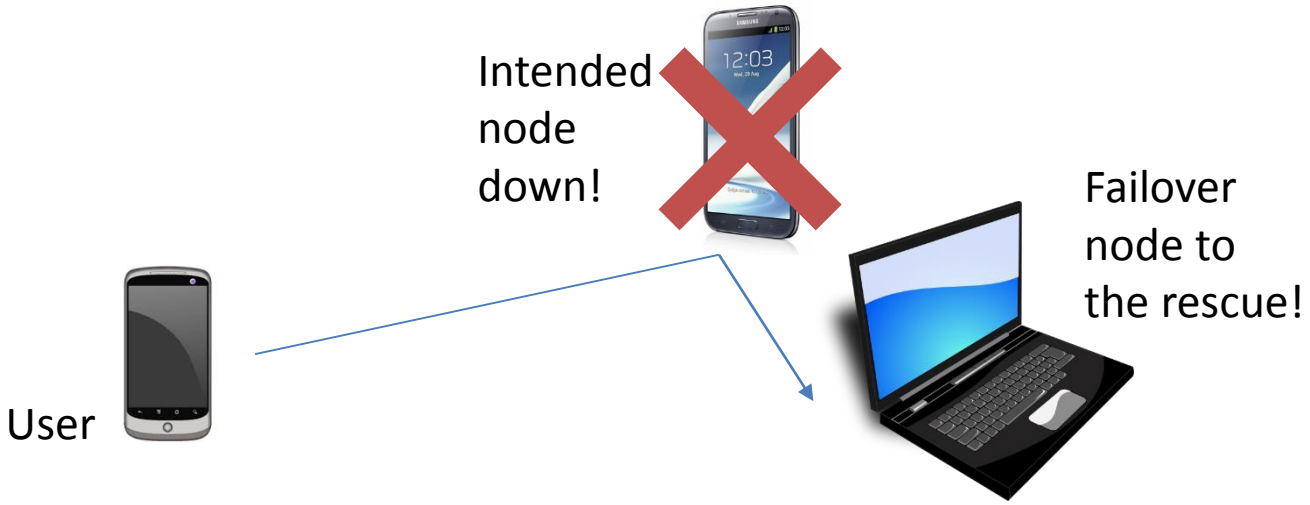
DNS Server

- Fairly Standard BIND9 server
- Lots of scripting to setup naming architecture

Name	Type	Value	Description
clay.elec529	A	168.7.138.104	Main host record
m.clay.elec529	A	168.7.138.104	Permanent master record
1.clay.elec529	CNAME	adriana.elec529	First failover
n.clay.elec529	CNAME	node.elec529	Nth failover
clay.elec529	MX	clay.elec529	Mail, priority, 1
clay.elec529	MX	1.clay.elec529	Mail failover, priority 10

Dynamic DNS Failover

- nsupdate to make changes without reloading server
- /etc/ppp/ip-up runs nsupdate to change ips
- API provided to Android for other changes
  - E.g. battery based failover



VPN Server

- PPP, xl2tpd, and openswan (ipsec)
- Fairly Standard
- Small caveats
  - VPN end point on network (routing loop)
  - 1 to 1 NAT

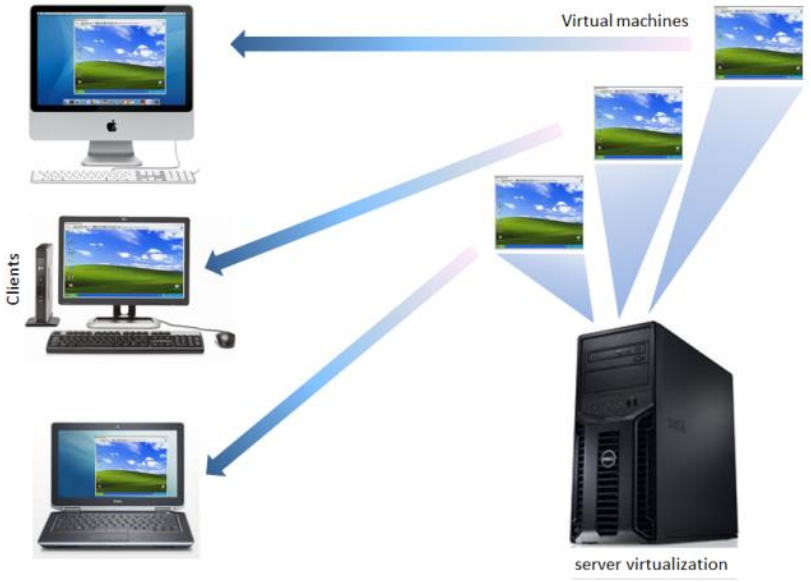
VM Test Environment

- Not enough phones
- Spin up VMs *on same IPs*
- Test JES, ssh, etc.

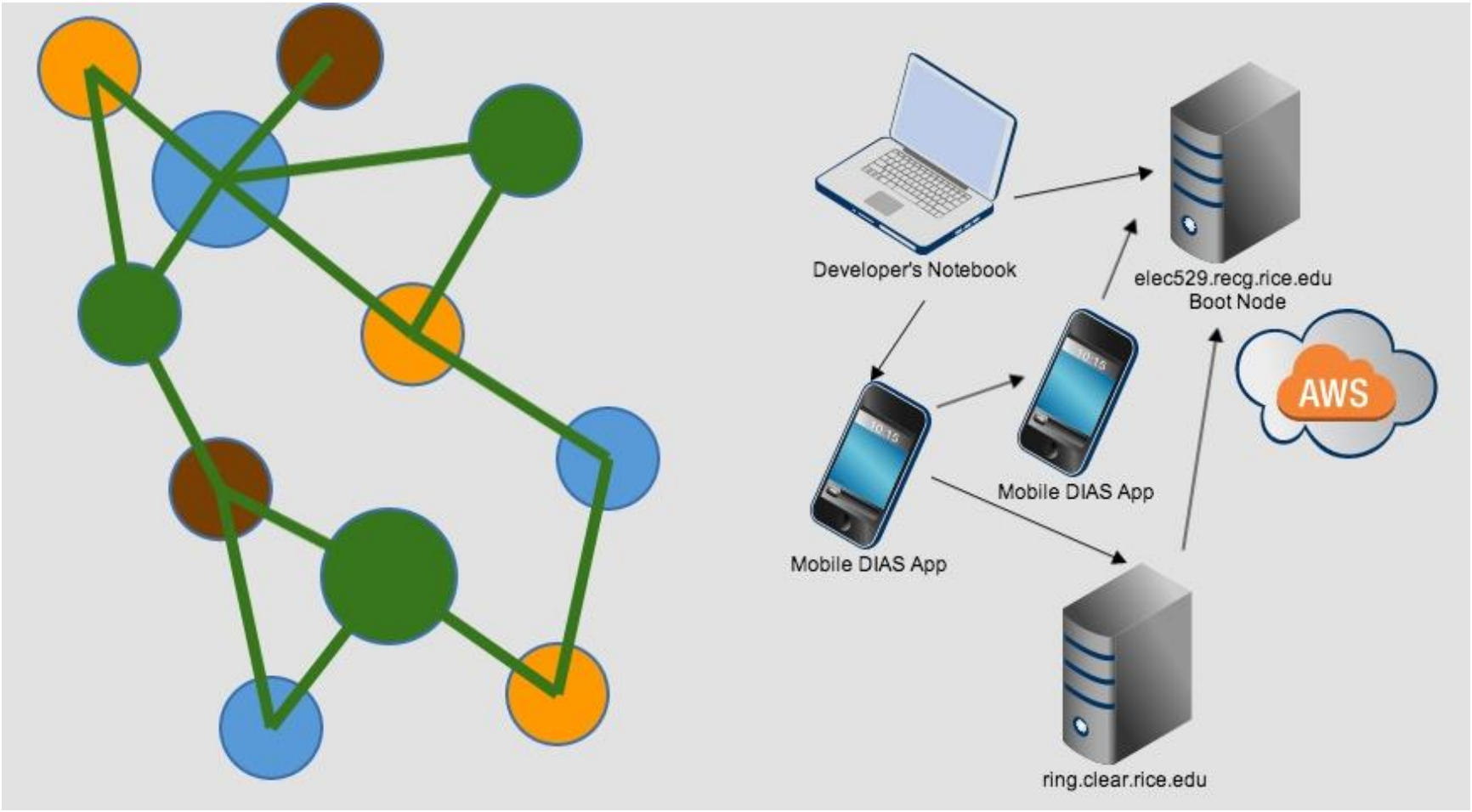
adriana.elec529.recg.rice.edu

ellis.elec529.recg.rice.edu

clay.elec529.recg.rice.edu



DIAS Overview



DIAS: WEB SERVER

DIAS: Web Server

- Nano HTTPD
  - Small
  - Single Java File
  - Embedded Systems
  - Multi-Threaded – 5 Threads
  - Port 8085
  - Modified to conform to Dias Service Interface
  - Modified to implement Dias Logging Service

DIAS: Web Server

jetty://

- Serverlet Engine
- Support Java EE
- Used NIO for I/O
- For Large Number but Not Busy Connection
- Small Memory Occupation
- Flexible

Apache Tomcat

- Serverlet Engine
- Support Java EE
- Used BIO for I/O
- For Small Number but Very Busy Connection
- Large Memory Occupation
- Integration

DIAS: Web Server

Jetty Introduction

- A Web server and javax.servlet container, plus support for SPDY, WebSocket, OSGi, JMX, JNDI, JAAS and many other integrations.

- These components are open source and available for commercial use and distribution

Eclipse Foundation

A not-for-profit, member supported corporation that hosts the Eclipse projects and helps cultivate both an open source community and an ecosystem of products and services.

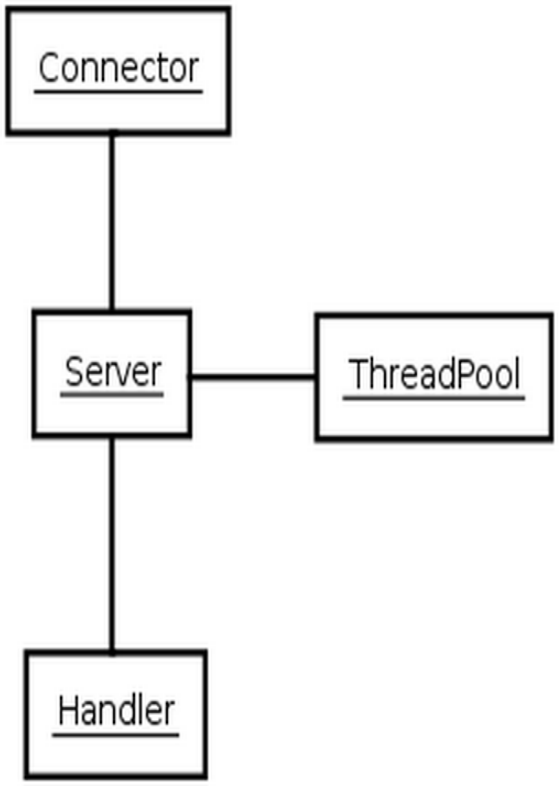
http://www.eclipse.org/jetty/

Powered

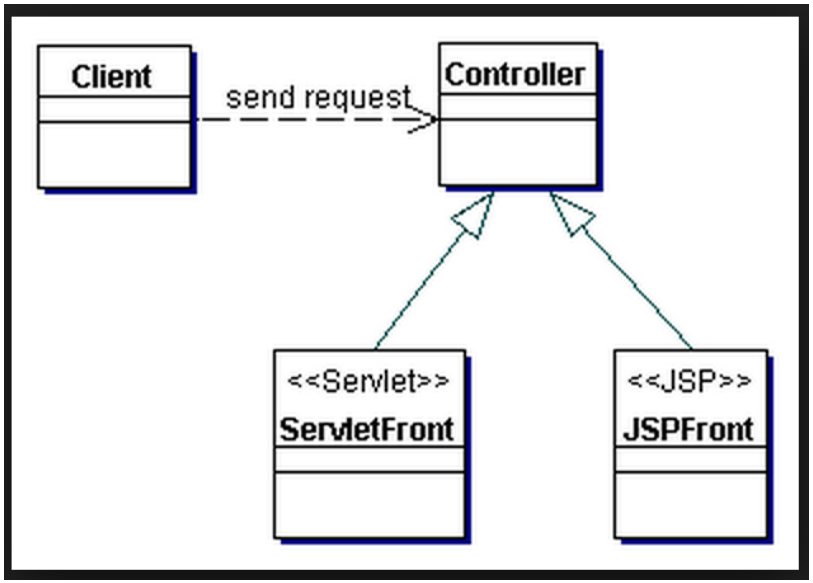
- Large clusters
  - Yahoo! Hadoop Cluster
- Cloud computing
  - Google App Engine
- SaaS
  - Yahoo! Zimbra
- Application Servers
  - Apache Geronimo
- Frameworks
  - Google Web Toolkit
- Devices
  - Android Mobile OS

DIAS: Web Server

The Structure of Jetty – View From 20000 Feet

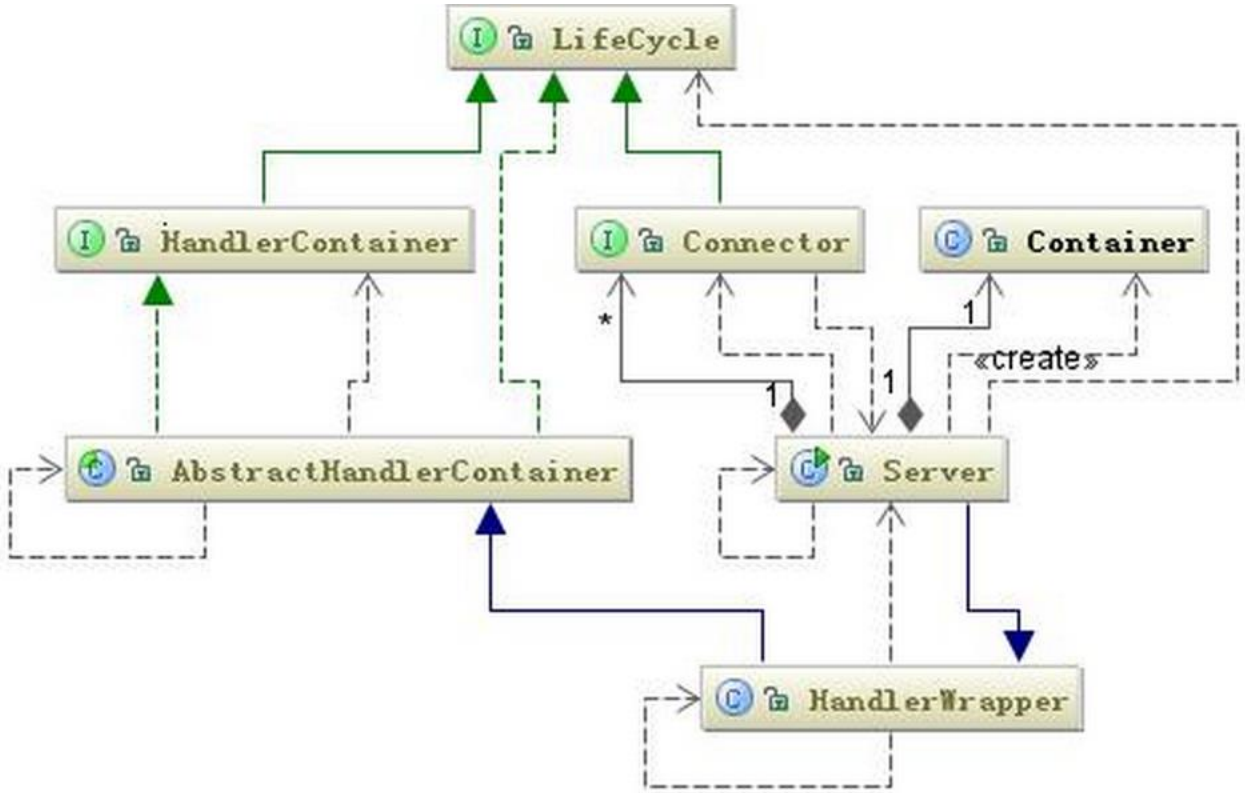


- Controller
  - Connector
  - Client
  - Queue
  - Thread
- View
  - JSP Front
  - Mobile Phone



DIAS: Web Server

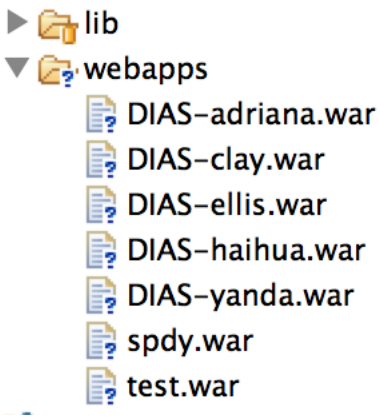
Model – Handler – Observable Design Patter



- **Implement Handler**
- **Add Handler**
- **Start Handler**

DIAS: Web Server

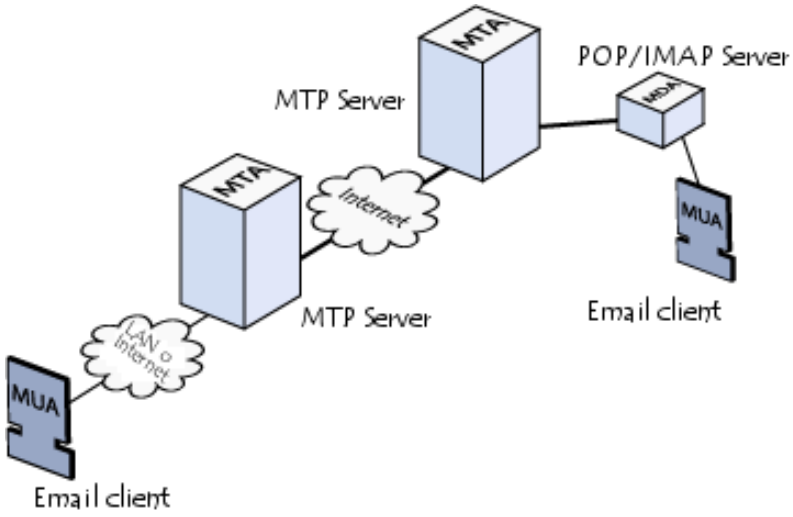
Embed Start

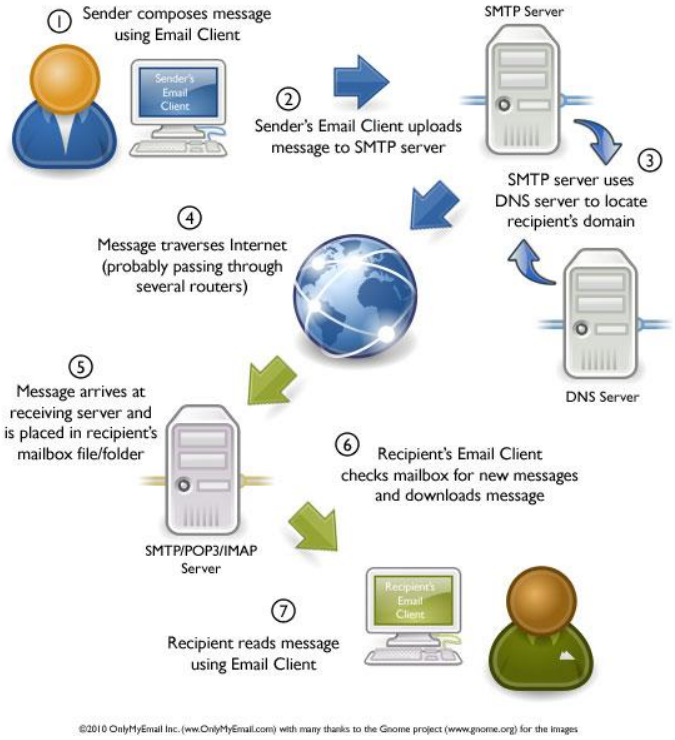



```
@Override
public void start() {
    // TODO Auto-generated method stub

    Handler handler = new AbstractHandler() {

        @Override
        public void handle(String arg0, Request arg1,
                HttpServletRequest request, HttpServletResponse servletResponse)
                        throws IOException, ServletException{
            // TODO Auto-generated method stub
            servletResponse.setContentType("text/html");
            servletResponse.setStatus(HttpServletResponse.SC_OK);
            servletResponse.getWriter().println("<h1>This is Yanda</h1>");
            ((Request) request).setHandled(true);

        }

    };


    this.server = new Server(port);
    server.setHandler(webApp);
    //server.setHandler(handler);
    server.setStopAtShutdown(true);
    try {
        server.start();
        //Log.e("JT", "started");
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}
```

DIAS: EMAIL SERVER

DIAS: Email Server

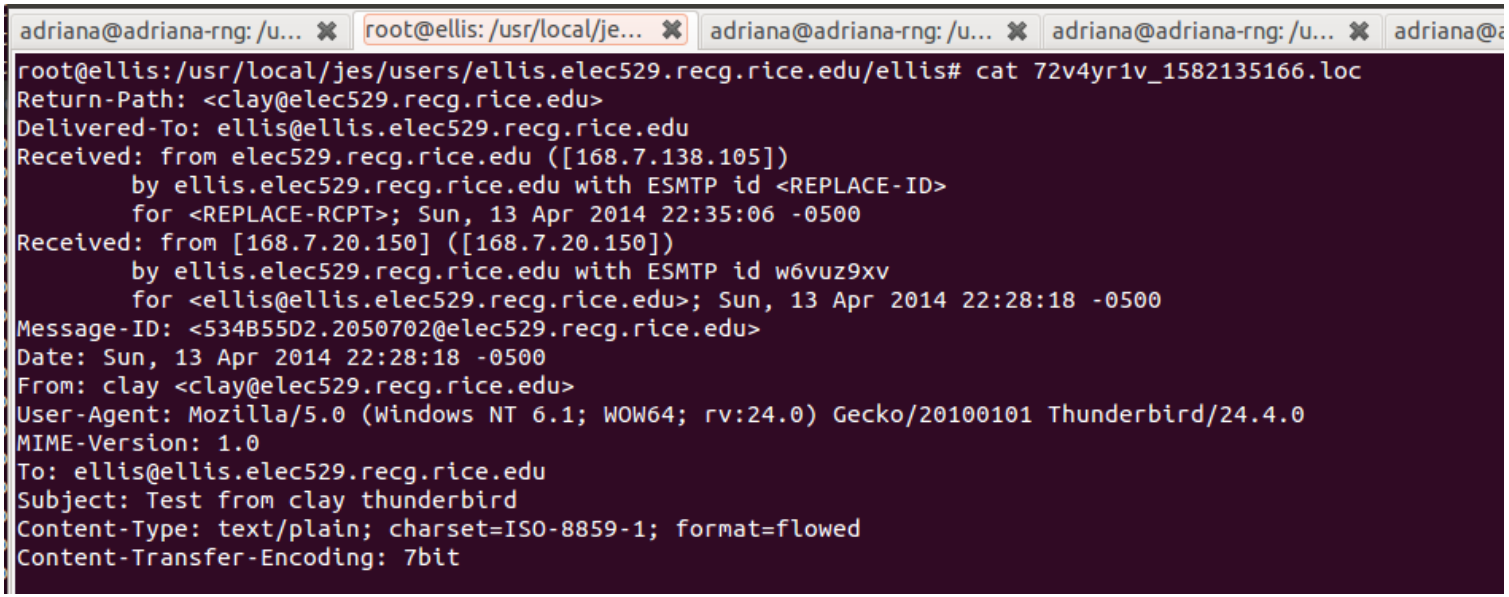




- MTA: Mail Transport Agent
- MTAs communicate with one another using the protocol SMTP
- MDA: Mail Delivery Agent
  - POP3 (Post Office Protocol)
  - IMAP (Internet Message Access Protocol)
- MUA: Mail User Agent

JES – Java Email Server

- JES is a multi-featured hybrid MTA/MDA server written in the java programming language
- Secure socket connections:
  - TLSv1, SSLv3
- Authentication mechanisms:
  - SASL PLAIN, LOGIN, CRAM-MD5, DIGEST-MD5, SCRAM-SHA1, GSSAPI
- Use filesystem to store emails



JES – Java Email Server

Mail.xml
- Domains
- Users
- Passwords

JES-DNS
- In-house software library to resolve IP and dns names

Logger
- JES allows commons-logging
- log4j

Configuration
- Mail.conf : Ports, # Threads, global security
- User.conf: local users and their pswd
- Log.conf – loads logj4 properties files

Service Wrapper
- Wrapper.conf
- Starting JES

Mail.sh
- Execute JES from command line
- JAVA_EXEC, JES_HOME

Security
- Java Cryptography Extension (JCE)
- JES Vault & Master Password
- JSSE keystore password
  - STARTTLS SMTP/POP3 extension

Configuration Manager BackEnd
- Domains, users and digest-MD5 realms
- File system or Database

DIAS: SSH & REPLICATION

DIAS: SSH Server

- Keys, paths, permissions, and env!
  - Very hard to get right.
  - Provides authentication for everything
- Based on OpenSSH
- Thread sets config, then monitors it.

DIAS: Data Replication (rsync)

- Keys, paths, permissions, and env!
  - Yes, again. This was hard. And ridiculous.
- Runs over SSH
  - Public Key Authentication
- Very efficient
  - Diffs files, and only sends modified blocks
- Two Modes: full and sparse replication

PEER TO PEER

Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems

Antony Rowstron[1] and Peter Druschel[2⋆]

[1] Microsoft Research Ltd, St. George House,
Guildhall Street, Cambridge, CB2 3NH, UK.
antr@microsoft.com
[2] Rice University MS-132, 6100 Main Street,
Houston, TX 77005-1892, USA.
druschel@cs.rice.edu

Pastry is a generic p2p content location and routing system based on a self-organizing overlay network of nodes connected via the Internet.

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Pastry is completely decentralized, fault-resilient, scalable, and reliably routes a message to the live node with a nodeId numerically closest to a key.

```
Finished creating new node Clayton PastryNode[SNH: <0x436C61..>/water.clear.rice
.edu/128.42.208.6:18010]
P2P.onConnected()
Clayton P2P.onPeerJoined - Ellis
Clayton P2P.onPeerJoined - Adriana
P2P.sendMessage() from Clayton to Clayton
Received from node Clayton. Data:

Received from node Ellis. Data:

P2P.sendMessage() from Clayton to Adriana
Received from node Adriana. Data:

P2P.sendMessage() from Clayton to Ellis
```

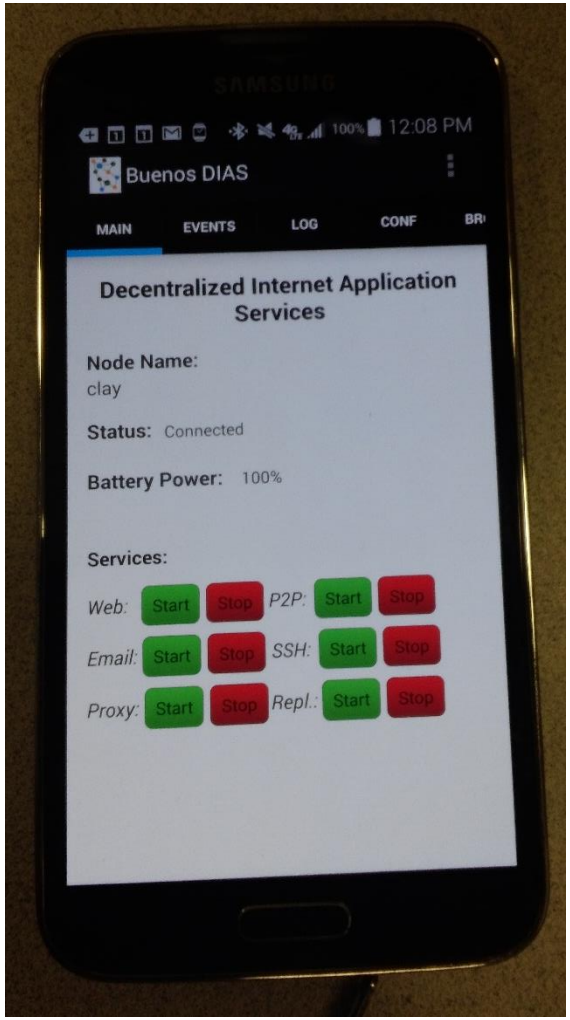
```
Ellis P2P.onPeerJoined - Clayton
Finished creating new node Ellis PastryNode[SNH: <0x456C6C..>/ring.clear.rice.ed
u/128.42.208.5:18011]
P2P.onConnected()
Ellis P2P.onPeerJoined - Adriana
P2P.sendMessage() from Ellis to Clayton
P2P.sendMessage() from Ellis to Adriana
Received from node Clayton. Data:

P2P.sendMessage() from Ellis to Ellis
Received from node Ellis. Data:

Received from node Adriana. Data:
```

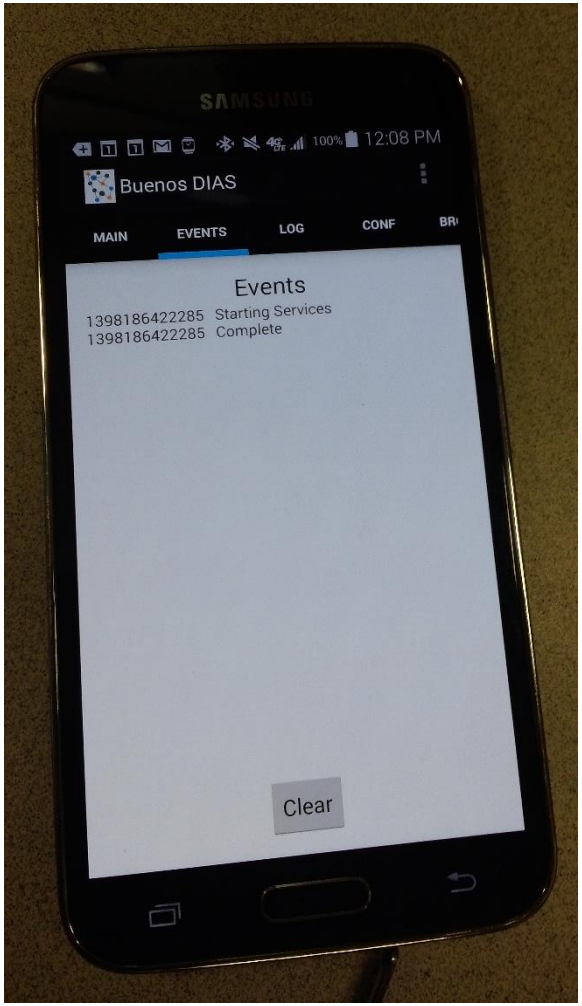
```
Adriana P2P.onPeerJoined - Ellis
Adriana P2P.onPeerJoined - Clayton
Finished creating new node Adriana PastryNode[SNH: <0x416472..>/sky.clear.rice.e
du/128.42.199.52:18013]
P2P.onConnected()
Received from node Clayton. Data:

P2P.sendMessage() from Adriana to Clayton
Received from node Ellis. Data:

P2P.sendMessage() from Adriana to Adriana
Received from node Adriana. Data:

P2P.sendMessage() from Adriana to Ellis
```

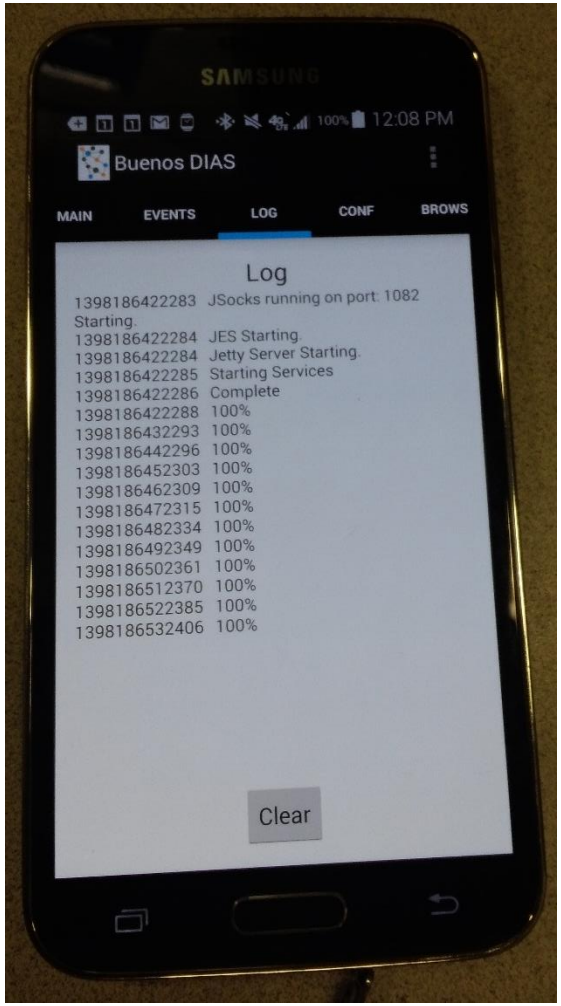
Pastry can be used for applications like file sharing, file storage, group communication, and naming systems.

ANDROID APP & DEMO

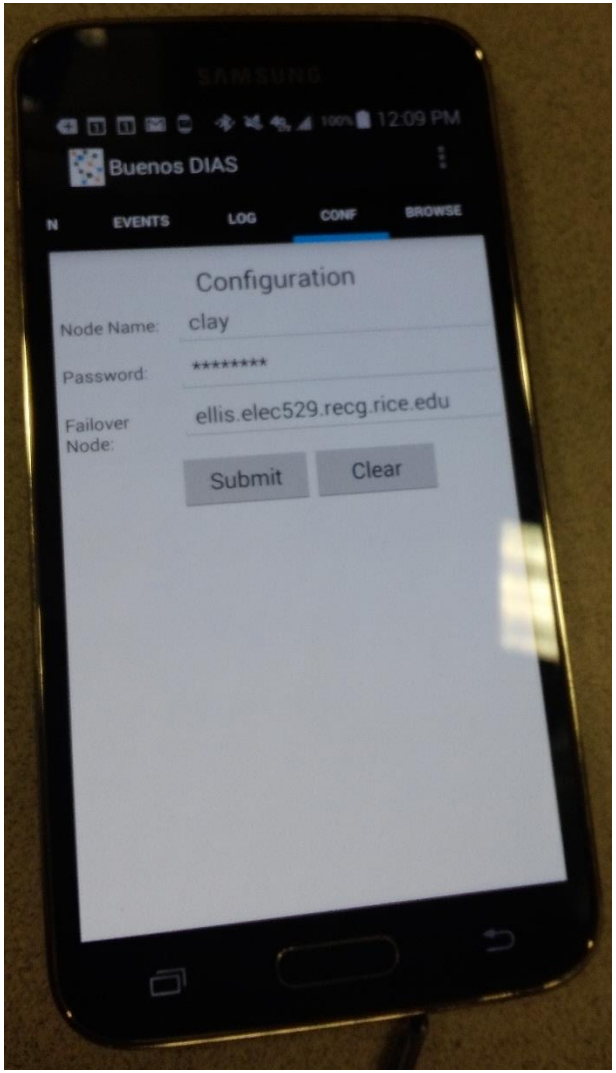
DIAS Android Application

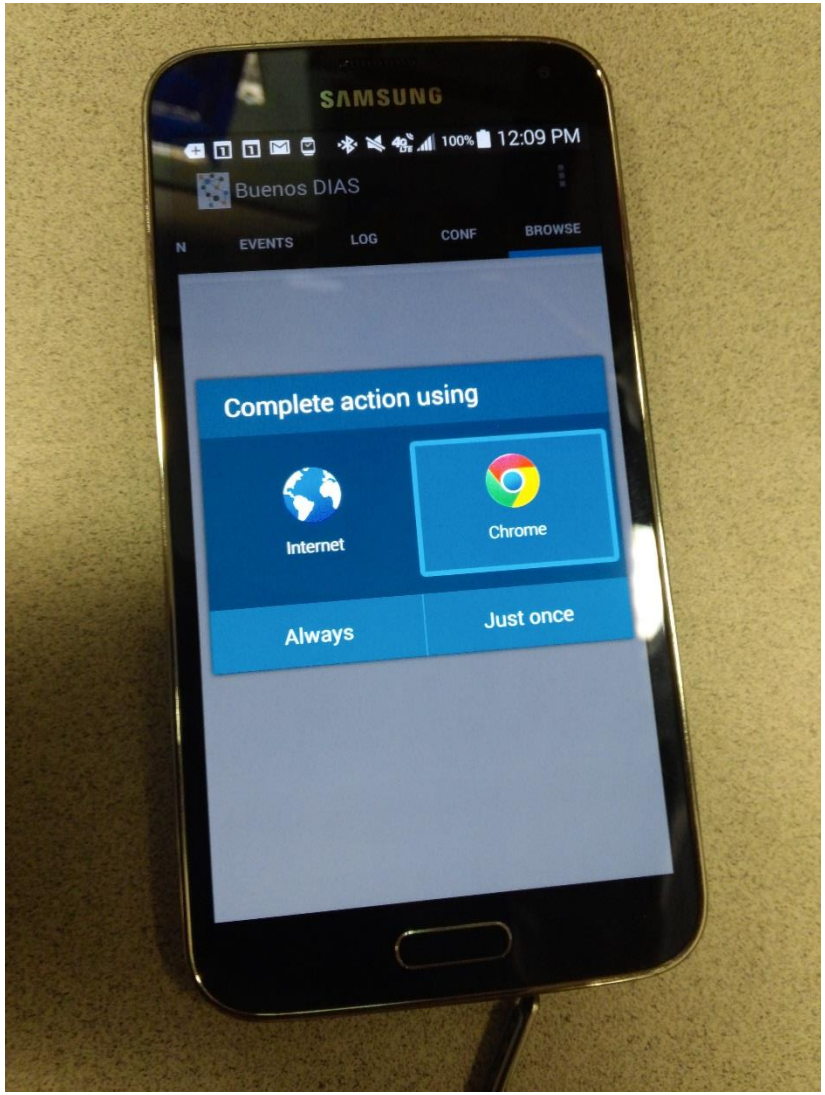






DIAS Android Application



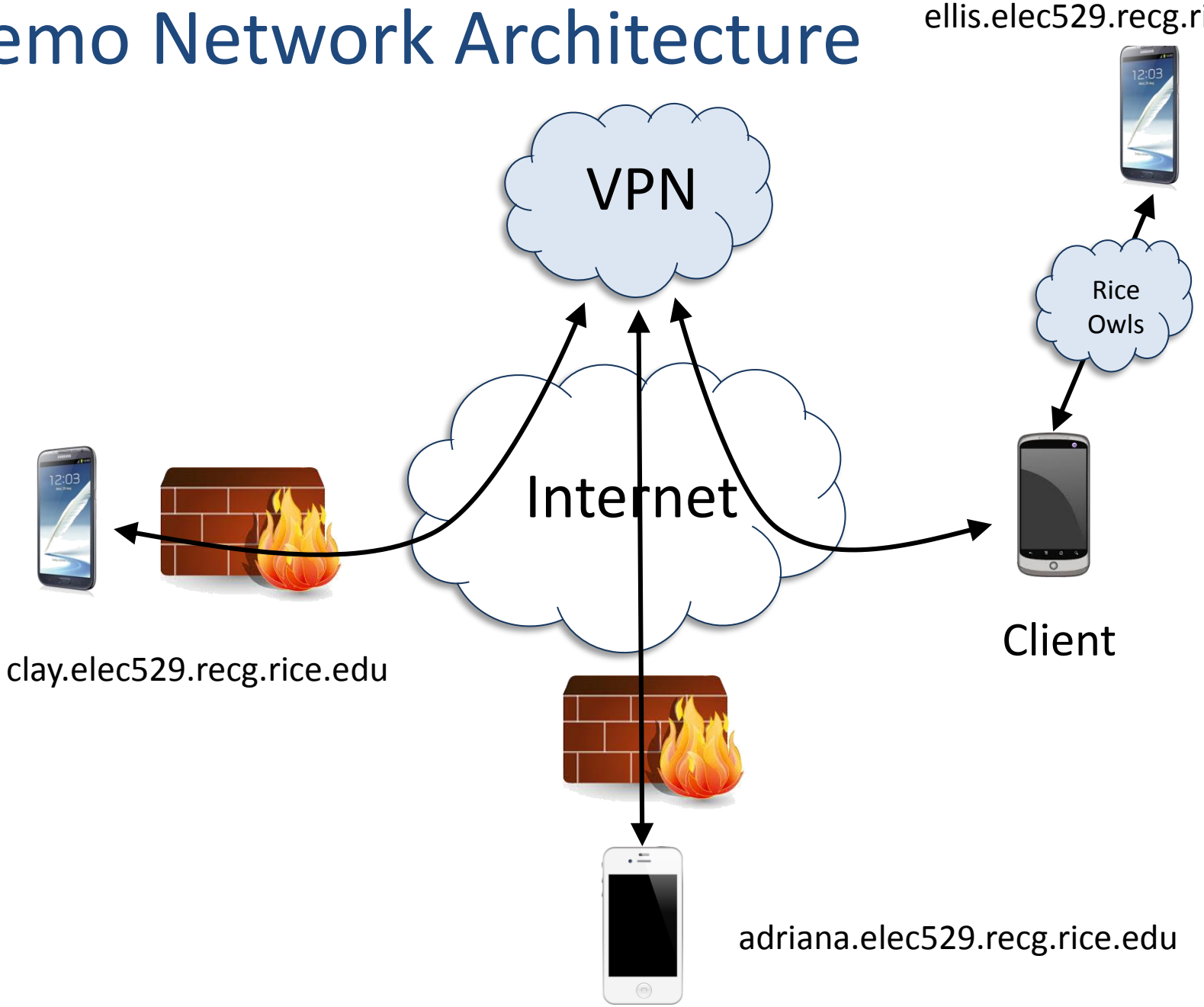


DEMO

Experiments

- Testing of individual services
  - Jetty, Kirium, etc.
- DNS and Proxy Testing
- Nano HTTPD Testing
- Email Testing
- Battery Tests
  - Galaxy S4 Running DIAS
  - Two Hour Tests:
    - Regular Usage 5%
    - Streaming 2.35 GB Data 28%
    - DIAS Regular Usage with proxy 10%
- Replication: Rsync and SSH Testing

Demo Network Architecture



CONCLUSIONS

Summary

- Build a first generation decentralized architecture which supports typical personal communication and applications including:
  - Email
  - Web Pages
- Benefits:
  - Security, Privacy, Resilience, Cost, and Power
- Challenges:
  - Redundancy, Uptime, Failover, Battery Management
- Development and Integration with Android Application

Forward Work

- P2P for transport
  - Could eliminate VPN
  - Could use peer for browsing service
- P2P for replication
  - Push files to peers not running rsync
- Email server running on a different port
  - Users don't have to have root, but email may be routed through DIAS servers that have 25 open.
- DIAS as a Platform-As-A-Service Model.
- Security