Microsoft® Research

# Faculty Summit

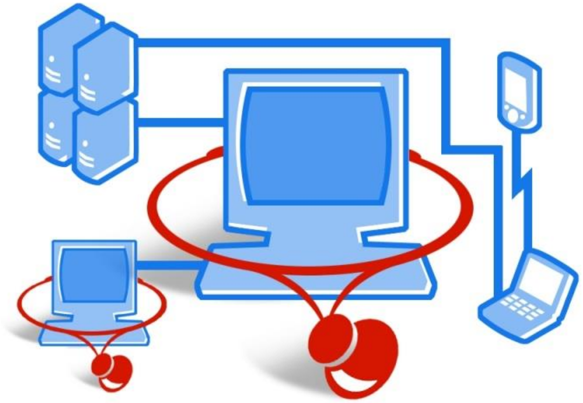
10

YEAR ANNIVERSARY

# Mobile Assistance Using the Internet

# The MAUI Project

Victor Bahl, Microsoft Research



Joint work with Aruna Balasubramanian (Intern, UMASS), Ranveer Chandra, Dae-Ki Cho (Intern, UCLA), Eduardo Cuervo Laffaye (Intern, Duke), Shravan Rayanchu (Wisconsin, intern), Stefan Saroiu, Alec Wolman

# Partners

Microsoft Research

The Maui Project

Carnegie Mellon

The Cloudlets Project

THE · UNIVERSITY · OF · CALIFORNIA · LET THERE BE LIGHT UCLA

The Guardian Phone Project

USC

The Language Support Project

# Sad Reality of Mobile Computing

Hardware limitations

– vs. static elements of same era (desktops, servers)

– weight, power, size constraints

– CPU, memory, display, keyboard …

W

**True 15+ years ago (early 1990s)**

- **huge hardware and wireless networking improvements since but deep essentials still the same. Will the same slide will be true in 2020?**

– intermittent connectivity

– may cost real money, require service agreements

Finite energy source

– actions may be slowed or deferred

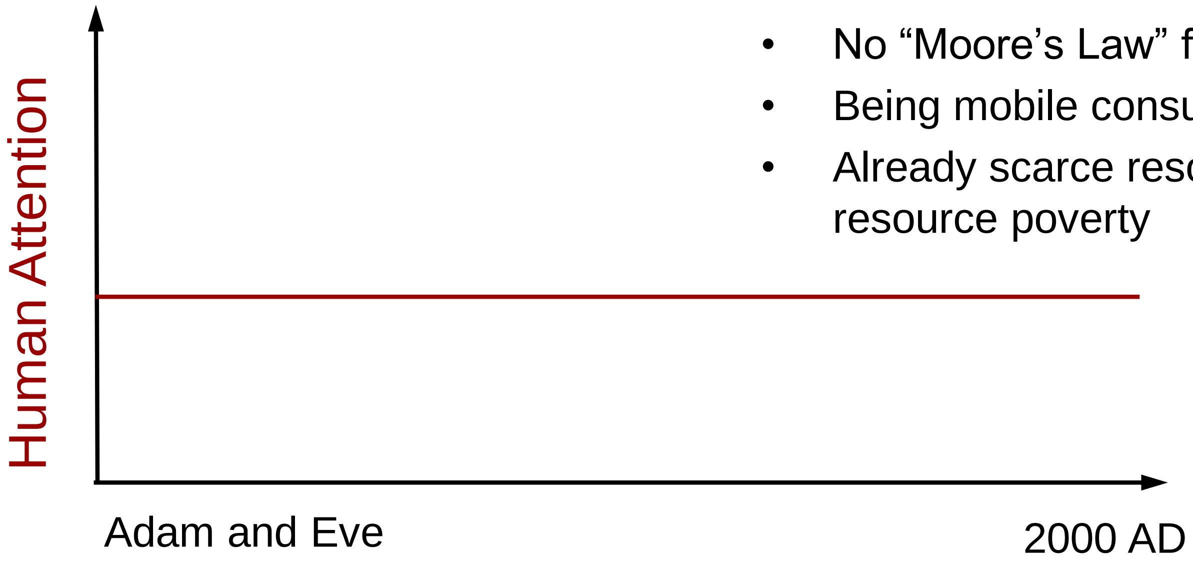
– wireless communication costs energy

How can we enable resource-rich mobile computing?

Courtesy. M. Satya, CMU

# Why Resource Poverty Hurts

- No “Moore’s Law” for human attention
- Being mobile consumes human attention
- Already scarce resource is further taxed by resource poverty



Reduce demand on human attention

- Software computing demands not rigidly constrained
- Many “expensive” techniques become a lot more useable when mobile

Some examples

- Machine learning, activity inferencing, context awareness
- Natural language translation, speech recognition, …
- Computer vision, context awareness, augmented reality
- Reuse of familiar (non-mobile) software environments

…

Clever exploitation needed to deliver these benefits

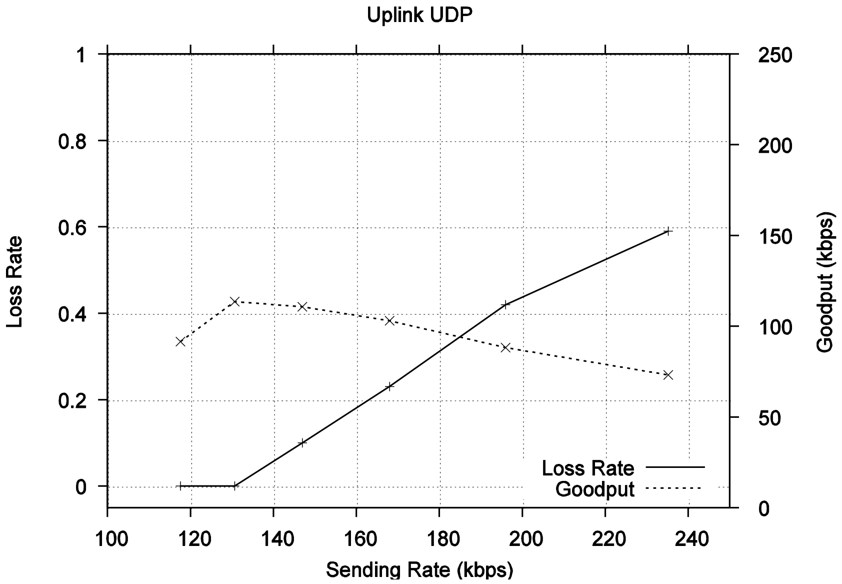
Vastly superior mobile user experience

# Scenarios We Want to Enable

- *Augmented Reality*
  - Mobile analyzes and utilizes data it senses
  - Example application: Helping patients with memory loss
- *Corrective Human Behavior*
  - Mobile analyzes & utilizes data that the user is generating
  - Example application: Correcting incorrectly spoken fact
- *Influencing Actions through Predictions*
  - Mobile analyzes & utilizes data that the user is generating AND that it is sensing
  - Example application: “Take a left turn ahead”
- *Mobile Games*
  - Xbox LAN parties, without the need to lug Xboxes around

# Is Cloud Computing the Answer?

End-to-end latency hurts interaction quality (crisp interaction essential for low demand on human attention)



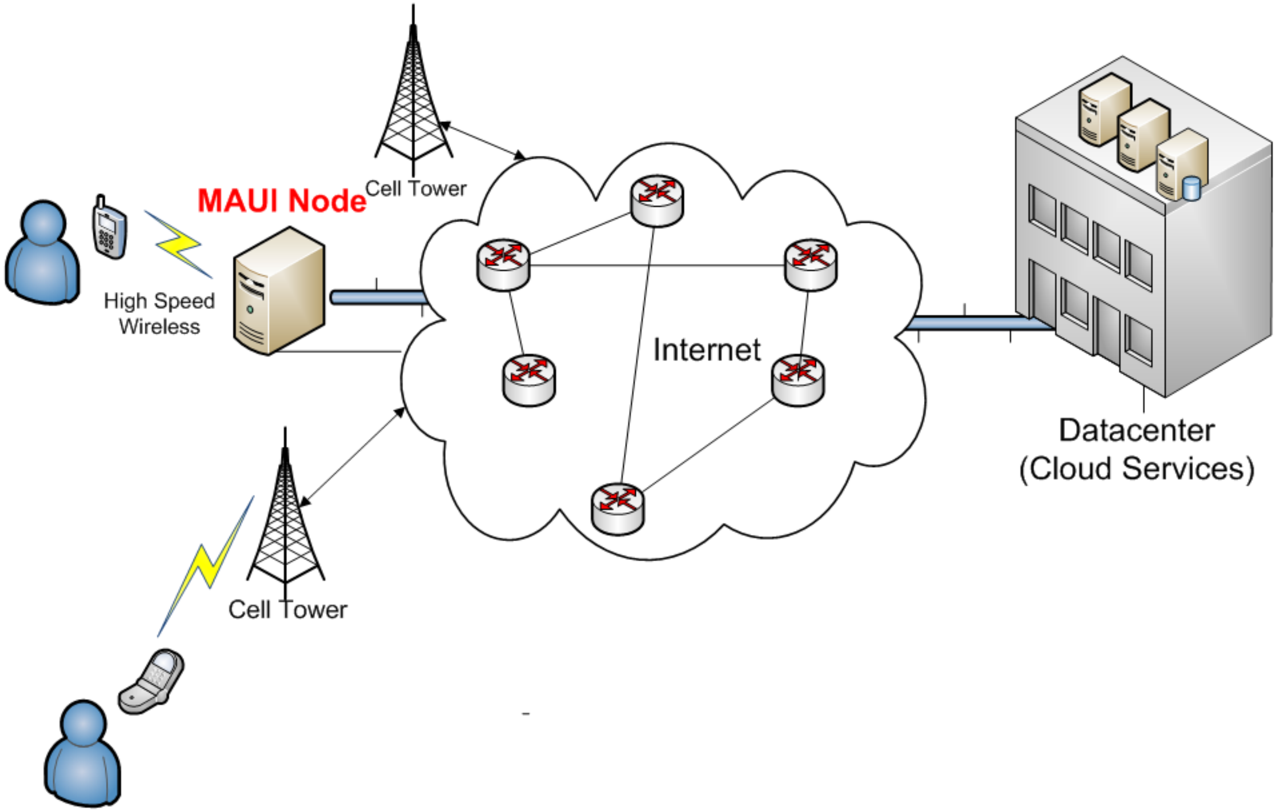
CDMA EV-DO network

High loss rate and low throughput severely limits the scope of cloud  services

We propose to push the “cloud” to within a few meters of the mobile user

# The MAUI System

A resource rich infra-structure computing device with high-speed Internet connectivity to the cloud that a mobile device can use to augment its capabilities and enable applications that were previously not possible.

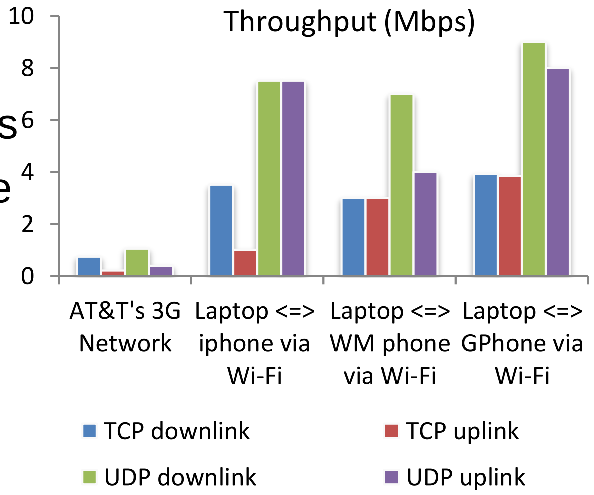


# The MAUI Node

A resource rich infra-structure computing device with high-speed Internet connectivity to the cloud that a mobile device can use to augment its capabilities and enable applications that were previously not possible

## *Basic Hardware Capabilities*

1. Ample and expandable CPU, memory and storage resources
2. Low latency, high bandwidth connection to the mobile device
3. Low latency, high bandwidth connection to the Internet
4. Physically secure



## *Basic Software Capabilities*

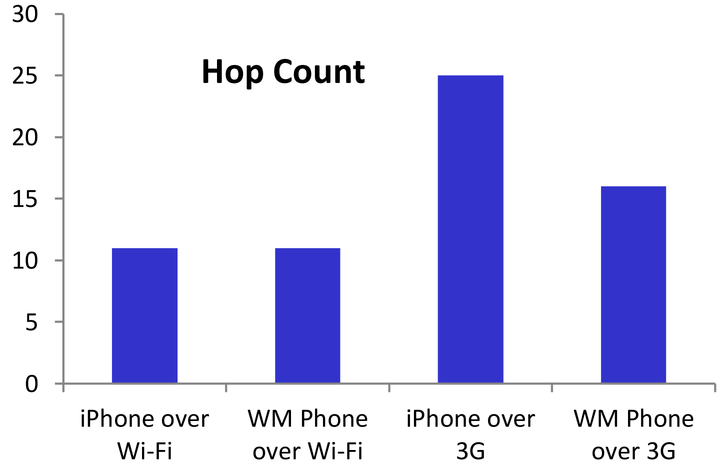
1. Trustworthy (established reputation)
2. Capable of running Virtual Machines
3. Runs lightweight discovery/capability/connectivity protocols
4. Remotely manageable (no need for an on-premises IT person )

# The case for MAUI: Latency

### iPhone via Wi-Fi : 11 hop

Wi-Fi -> 209.85.225.99

```
1.  (10.0.2.1) 8.513 ms 8.223 ms 9.365 ms
2.  (141.212.111.1)  0.913 ms  0.606 ms  0.399 ms
3.  (192.122.183.41)  11.381 ms  6.054 ms  5.975 ms
4.  (192.12.80.69)  7.038 ms  7.353 ms  7.026 ms
5.  (198.108.23.12)  12.525 ms  13.027 ms  12.619 ms
6.  (198.110.131.78)  12.715 ms  9.424 ms  9.315 ms
7.  (216.239.48.154)  9.974 ms (209.85.250.237)  10.295 ms
    (216.239.48.154)  9.405 ms
8.  (72.14.232.141)  19.308 ms  22.249 ms  23.312 ms
9.  (209.85.241.35)  32.987 ms  22.708 ms (209.85.241.27)  124.588 ms
10. (72.14.239.18)  22.256 ms (209.85.248.106)  29.154 ms
    (209.85.248.102)  21.635 ms
11. (209.85.225.99)  19.973 ms  21.930 ms  21.656 ms
```



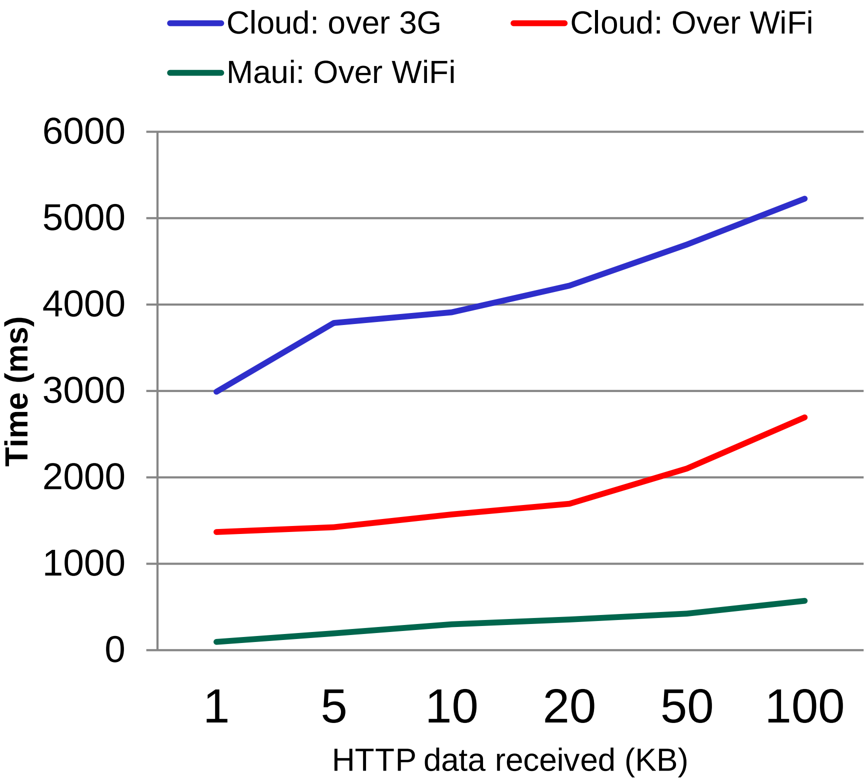
*traceroute* to 209.85.225.99 (one of the server IPs of www.google.com)

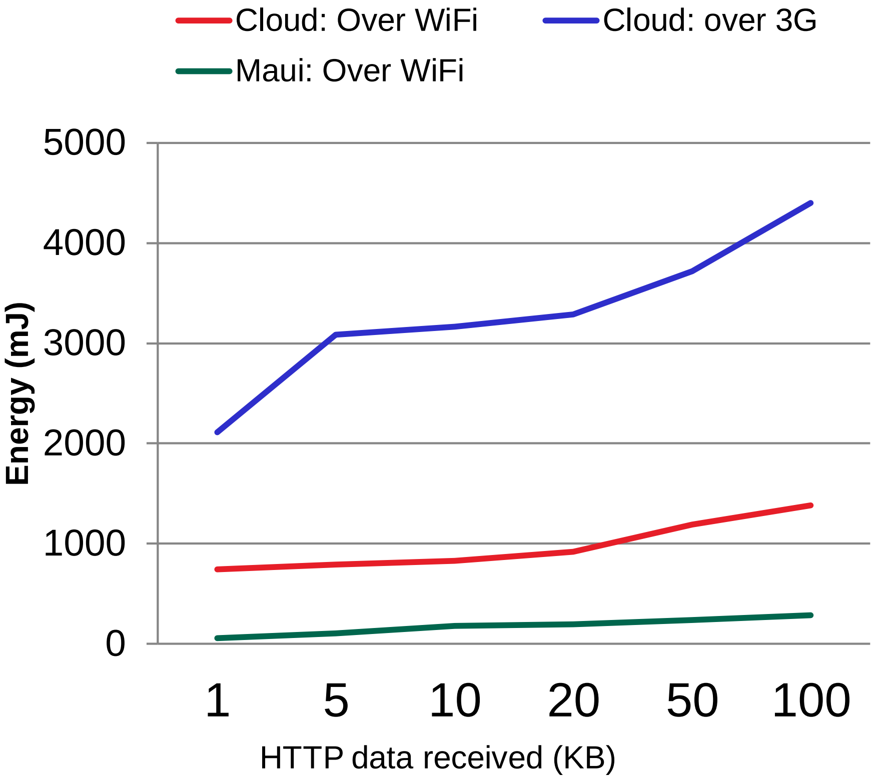
### iPhone via 3G : 25 hop

3G -> 209.85.225.99

```
1.  * * *
2.  (172.26.248.2)  414.197 ms  698.485 ms  539.776 ms
3.  (172.16.7.82)  1029.853 ms  719.595 ms  509.750 ms
4.  (10.251.11.23)  689.837 ms  669.340 ms  689.739 ms
5.  (10.251.10.2)  509.781 ms  729.746 ms  679.787 ms
6.  (10.252.1.7)  719.652 ms  760.612 ms  788.914 ms
7.  (209.183.48.2)  689.834 ms  599.675 ms  559.694 ms
8.  (172.16.0.66)  539.712 ms  809.954 ms  689.547 ms
9.  (12.88.242.189)  589.857 ms  1129.848 ms  709.784 ms
10. (12.122.138.38)  589.699 ms  1009.723 ms  769.808 ms
11. (12.122.138.21)  669.690 ms  529.758 ms  699.965 ms
12. (192.205.35.222)  699.569 ms  979.769 ms  1489.869 ms
13. (4.68.19.190)  699.435 ms (4.68.19.126)  559.875 ms (4.68.19.62)  499.598
14. (4.69.136.149)  889.946 ms (4.69.136.141)  879.443 ms
    (4.69.136.145)  469.601 ms
15. (4.69.132.105)  559.716 ms  539.733 ms  1219.982 ms
16. (4.69.132.38)  719.700 ms  659.613 ms  539.695 ms
17. (4.69.132.62)  549.752 ms  549.640 ms  800.128 ms
18. (4.69.132.114)  669.729 ms (4.69.140.189)  769.711 ms  959.663 ms
19. (4.69.140.193)  959.735 ms  979.674 ms  849.886 ms
20. (4.68.101.34)  649.609 ms  659.767 ms (4.68.101.98)  1119.996 ms
21. (4.79.208.18)  669.405 ms  629.574 ms (209.85.240.158)  1200.039 ms
22. (209.85.240.158)  769.538 ms (72.14.232.141)  729.505 ms
    (209.85.241.22)  719.715 ms
23. (209.85.241.22)  769.665 ms (209.85.241.35)  769.880 ms  859.536 ms
24. (209.85.241.29)  589.710 ms (66.249.95.138)  789.762 ms
    (209.85.248.106)  913.287 ms
25. (209.85.225.99)  716.000 ms (66.249.95.138)  1039.963 ms
    (72.14.239.18)  899.607 ms
```

# The Case for MAUI:  Energy





**Communications Time**

**Communications Energy**

# So What Does MAUI Give Us?

## Mobile Device Perspective

- Better application performance
  - Human attention management
- New application / behavior enablement. New scenarios
- Improved latency management
- Extensible computing horsepower
- Efficient spectrum usage - improved congestion / bandwidth management
- Improved energy management – longer battery life

# What’s Missing?

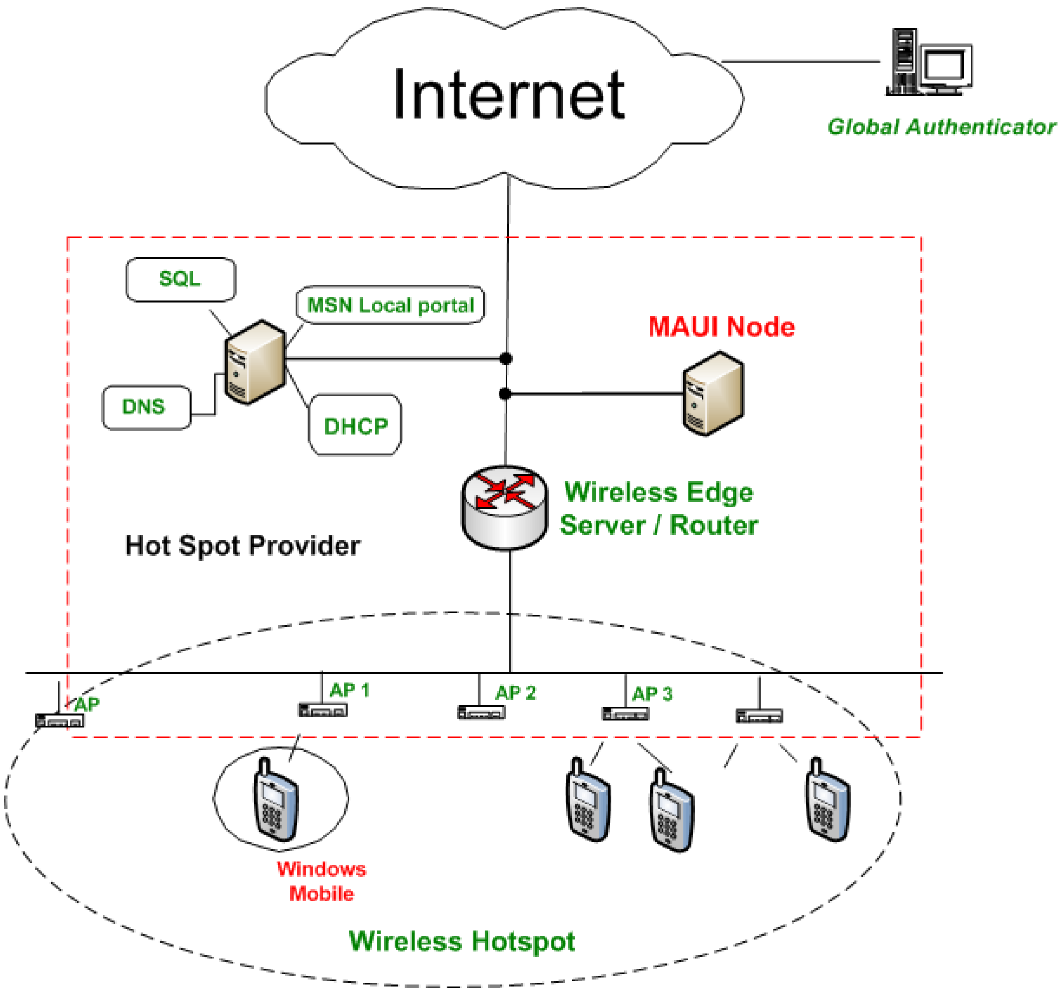
- Transparent leveraging of infrastructure resources
- New applications and usage models that leverage infrastructure
  - Synthesis of new technologies into seamless whole
- Fallback strategies when infrastructure unavailable
- New business models and new partnerships

The Gestalt Principle

“the whole is greater than the sum of its part”

# Possible Deployment Scenario

A MAUI enabled wireless hotspot



# MAUI Software Components

- Programming environment for developing mobile application to simplify the task for partitioning

- A Runtime environment for the *SmartPhone* that makes dynamic decisions based on current characteristics

- A Secure environment for running offloaded code in the infrastructure

# Programming Environment

## Application Partitioning Goals

- Automatic partitioning of the application between the mobile device and the infrastructure

- Minimize mandatory developer involvement

- Configurable for improved performance

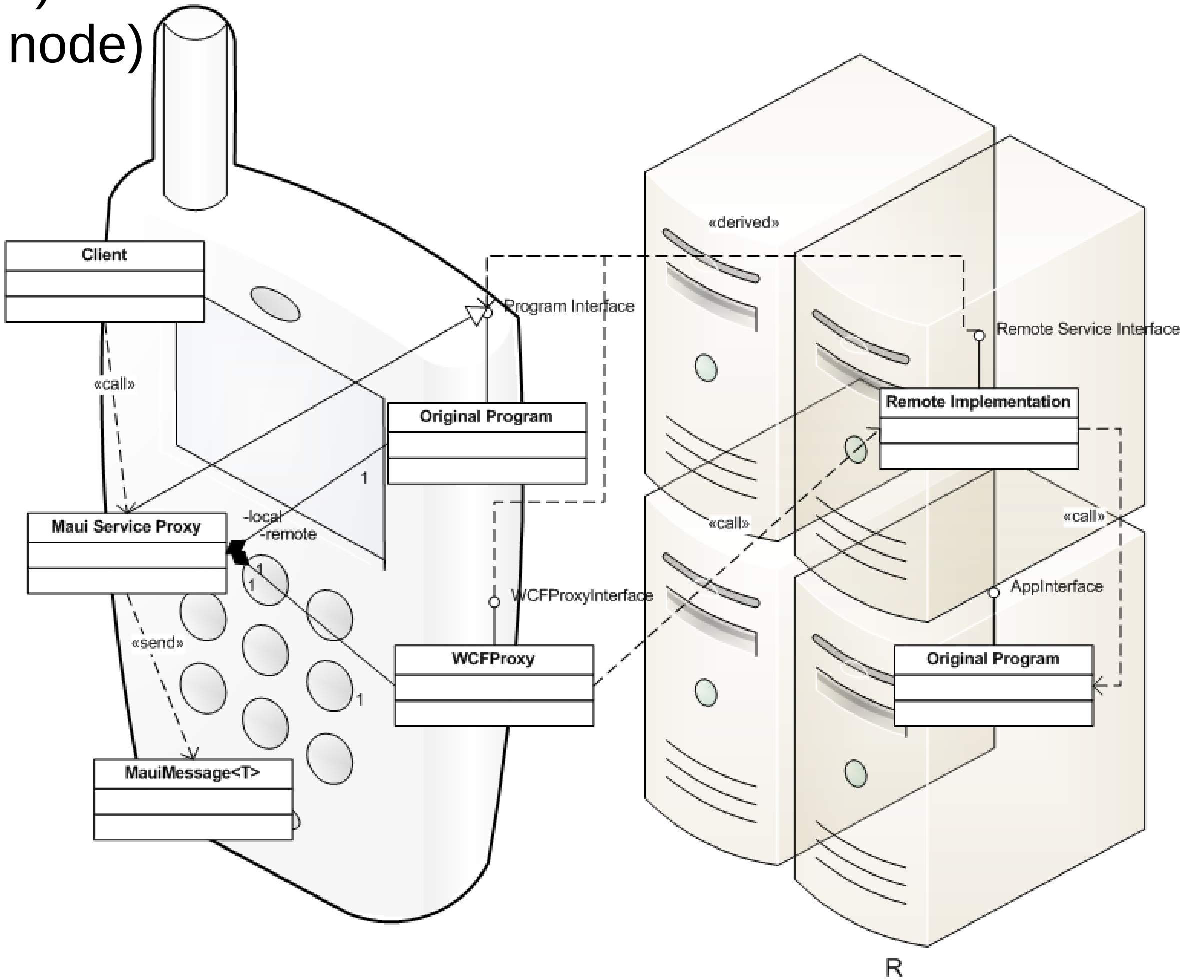
### Partitioning

  - Class and method level partitioning
  - State synchronization
    - Communication through deltas
  - Built as a layer on Windows Communications Foundation (WCF)

# Programming Framework

Primary Components

- Original Program
- Proxy/dispatcher (Mobile)
- Remote Service (MAUI node)

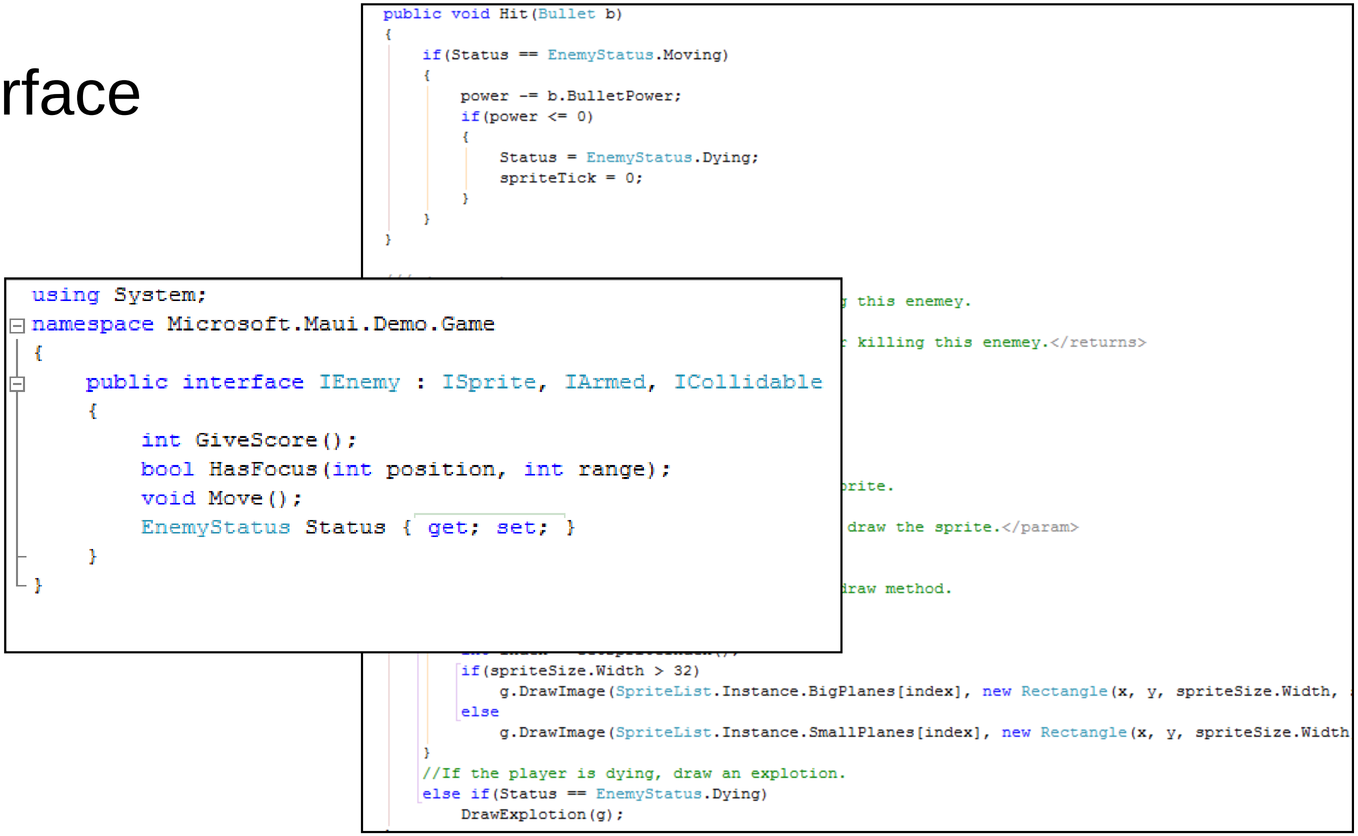


# *Mauizing* an Application

## Step 1: Extracting Interface

Original Program

Program Interface



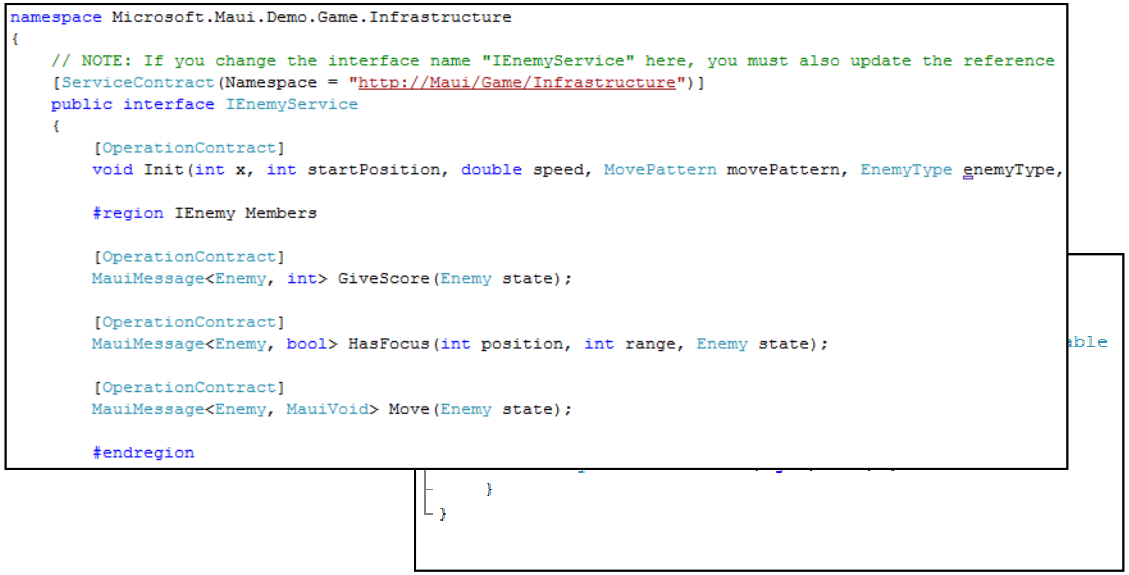
```
public void Hit(Bullet b)
{
    if(Status == EnemyStatus.Moving)
    {
        power -= b.BulletPower;
        if(power <= 0)
        {
            Status = EnemyStatus.Dying;
            spriteTick = 0;
        }
    }
}

                                                g this enemey.

                                                r killing this enemey.</returns>

                                                prite.

                                                 draw the sprite.</param>

                                                draw method.

        if(spriteSize.Width > 32)
            g.DrawImage(SpriteList.Instance.BigPlanes[index], new Rectangle(x, y, spriteSize.Width,
        else
            g.DrawImage(SpriteList.Instance.SmallPlanes[index], new Rectangle(x, y, spriteSize.Width
    }
    //If the player is dying, draw an explotion.
    else if(Status == EnemyStatus.Dying)
        DrawExplotion(g);
```

```
using System;
namespace Microsoft.Maui.Demo.Game
{
    public interface IEnemy : ISprite, IArmed, ICollidable
    {
        int GiveScore();
        bool HasFocus(int position, int range);
        void Move();
        EnemyStatus Status { get; set; }
    }
}
```

# *Mauizing* an Application

## Step 2: Add State synchronization

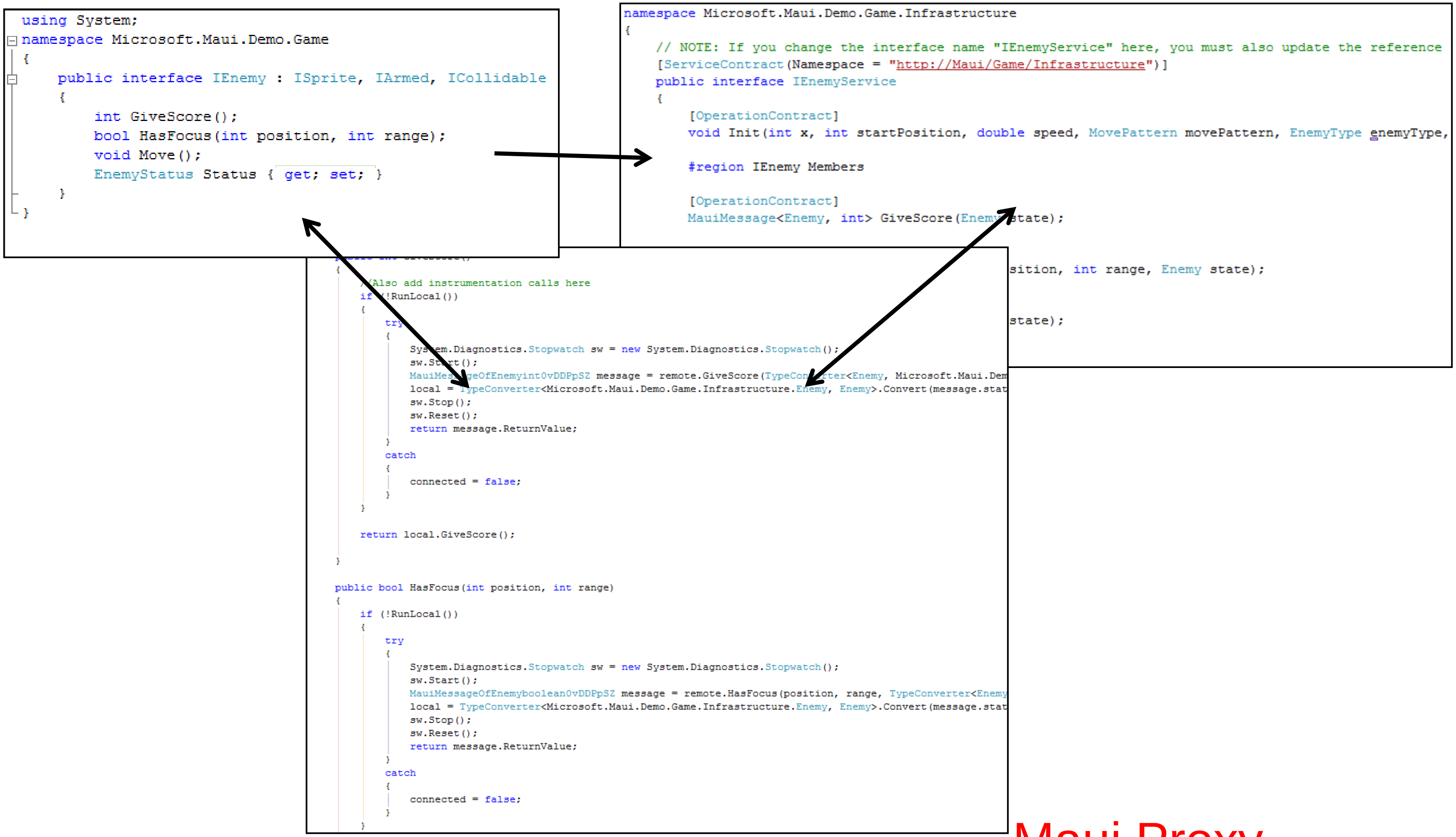
Remote Service Interface



```
namespace Microsoft.Maui.Demo.Game.Infrastructure
{
    // NOTE: If you change the interface name "IEnemyService" here, you must also update the reference
    [ServiceContract(Namespace = "http://Maui/Game/Infrastructure")]
    public interface IEnemyService
    {
        [OperationContract]
        void Init(int x, int startPosition, double speed, MovePattern movePattern, EnemyType enemyType,

        #region IEnemy Members

        [OperationContract]
        MauiMessage<Enemy, int> GiveScore(Enemy state);

        [OperationContract]
        MauiMessage<Enemy, bool> HasFocus(int position, int range, Enemy state);

        [OperationContract]
        MauiMessage<Enemy, MauiVoid> Move(Enemy state);

        #endregion
```

```
able
        }
}
```

# Step 3: A Maui Proxy / Dispatcher



```
using System;
namespace Microsoft.Maui.Demo.Game
{
    public interface IEnemy : ISprite, IArmed, ICollidable
    {
        int GiveScore();
        bool HasFocus(int position, int range);
        void Move();
        EnemyStatus Status { get; set; }
    }
}
```

```
namespace Microsoft.Maui.Demo.Game.Infrastructure
{
    // NOTE: If you change the interface name "IEnemyService" here, you must also update the reference
    [ServiceContract(Namespace = "http://Maui/Game/Infrastructure")]
    public interface IEnemyService
    {
        [OperationContract]
        void Init(int x, int startPosition, double speed, MovePattern movePattern, EnemyType enemyType,

        #region IEnemy Members

        [OperationContract]
        MauiMessage<Enemy, int> GiveScore(Enemy state);


                                                sition, int range, Enemy state);


                                                state);
```

```
    public int GiveScore()
    {
        //Also add instrumentation calls here
        if (!RunLocal())
        {
            try
            {
                System.Diagnostics.Stopwatch sw = new System.Diagnostics.Stopwatch();
                sw.Start();
                MauiMessageOfEnemyint0vDDPpSZ message = remote.GiveScore(TypeConverter<Enemy, Microsoft.Maui.Dem
                local = TypeConverter<Microsoft.Maui.Demo.Game.Infrastructure.Enemy, Enemy>.Convert(message.stat
                sw.Stop();
                sw.Reset();
                return message.ReturnValue;
            }
            catch
            {
                connected = false;
            }
        }

        return local.GiveScore();

    }

    public bool HasFocus(int position, int range)
    {
        if (!RunLocal())
        {
            try
            {
                System.Diagnostics.Stopwatch sw = new System.Diagnostics.Stopwatch();
                sw.Start();
                MauiMessageOfEnemyboolean0vDDPpSZ message = remote.HasFocus(position, range, TypeConverter<Enemy
                local = TypeConverter<Microsoft.Maui.Demo.Game.Infrastructure.Enemy, Enemy>.Convert(message.stat
                sw.Stop();
                sw.Reset();
                return message.ReturnValue;
            }
            catch
            {
                connected = false;
            }
        }
```

Maui Proxy

# Programming Framework

## Additional Features

- Attribute annotation describes application constrains
  - Allows better partitioning results
- Policy should make even naïve partitionings work
- Dynamic deployment and exposure of binaries
- Exposes built-in services (solving LP)

# Managing Energy through Application Partitioning……

Problem, Solution and Demo

# Energy Efficient Computing Using MAUI

Key Idea: Offload computation to infrastructure

Problem:

Which subset of methods should be executed at the infrastructure node?

Constraints

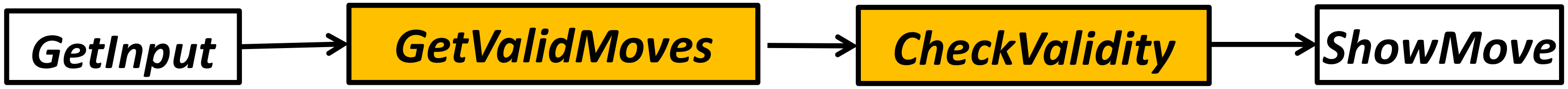
- Computation energy saving > Communication energy
- Application performance should not suffer

# MAUI Decision Engine: 3 Steps

- Step 1: Learn communication and computation energy for each method
  - Perform a per-device profiling *once*
  - Model energy utilization as a function of CPU cycles, amount of state transferred etc
- Step 2: Formulate the decision problem as an ILP and solve
  - Objective: minimize energy
- Step 3: Re-evaluate energy predictions and solve ILP periodically

# Demo Setup

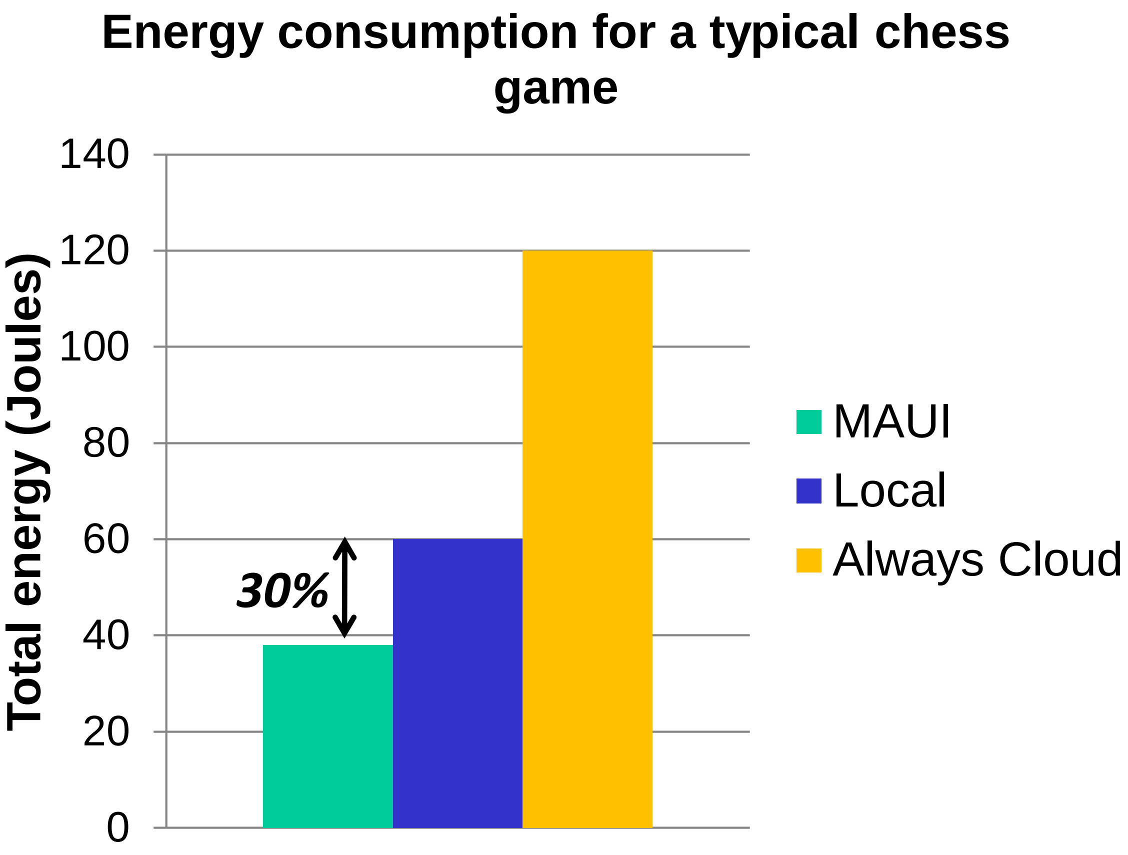
Chess Application



2 scenarios:

- Connect to a nearby MAUI node over WiFi
- Connect to the cloud computer (located in building 99) over 3G

# Energy Savings

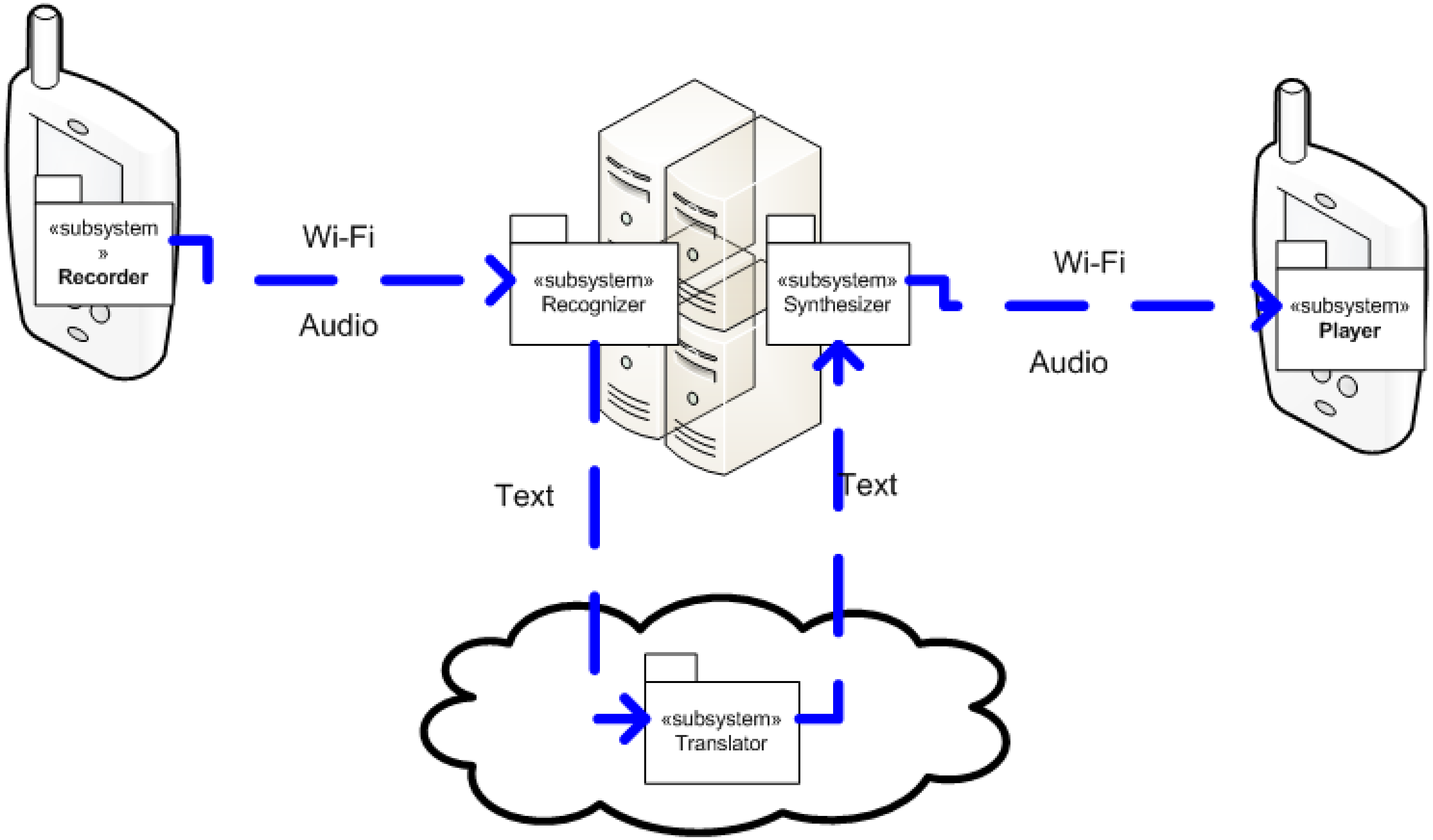


- Response time improved by 180 ms per move

# Making research relevant and interesting…

Real-time Voice Translator demo

# Voice Translator – Demo Setup



# MAUI Future: Leverage New Technologies

- Low latency high, bandwidth short range wireless: transfer rates:  600 Mbps (.11n),  several Gbps  (60 GHz)
- Low latency, high bandwidth connection to the Internet (the Cloud) enabled by: fiber to the home
- Non-geeky heads up displays, On-person Bluetooth devices
- Mobile IA32 chips (e.g. Atom) & multi-core mobile hardware with independent power-up for energy control
- High-capacity mobile storage (e.g. 16 GB flash)
- Cognitive Reasoning that incorporates  advances in
  - Speech Recognition
  - Natural Language Translation
  - Computer Vision
  - Context Aware Computing
  - Multi-radio peer localization and indoor location determination

**The metric of success is superior user experience**

# Thanks!

*http://research.microsoft.com/nrg/*

***The information herein is for informational purposes only and represents the current view of Microsoft Corporation as of the date of this presentation. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information provided after the date of this presentation. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS PRESENTATION.***