



UNIVERSITY OF  
CAMBRIDGE

# Network Architecture Research Considerations Or The Internet Conspiracy

**Jon Crowcroft (University of Cambridge)**

James Scott (xxx Research Cambridge)

Pan Hui (University of Cambridge)

Christophe Diot (Thomson)

Mothy Roscoe (ETH)

# Why we shouldn't do network arch?

- Liberally borrowed slides from Mothy Roscoe (ex cambridge, sprint, intel, berkeley, oz, now ETH)
  - Post-modernism (Wenjun Hu, Jon, Mothy)?
  - No to FIND/GENI? - n.b. my abstract's comments(\*)
- Experience
  - Architectures emerge (ANSA/Herbert)
  - Papers about them are usually post-hoc rationalisations
- First, general Arguments
- Then specific (Haggle) example...

# Why have an architecture?

- What does an Internet Architecture hope to achieve?
  - Interoperability across networks
  - Easier for applications to code to
  - Framework for providers to compete
- Does *any* Internet architecture really address these issues?

# What is a Network Architecture?

- Choose Paradigm
  - *Given Physical* constraint
    - Packet/circuit/new(e.g. multihop radio)?
  - Fundamentally is net a “*graph*”?
  - Are protocols/services “*layered*”?
- Choose Functional Decomposition
  - Trade between packet *header* and node
    - *Choose stateless or stateful (e2e v. hbh)*
    - *(e.g. change trade to include transport:*
    - *can do NAT, Header Compression and QoS/Flows)*
  - Are nodes different (host v. router)?
  - Choose Packet Format(s)

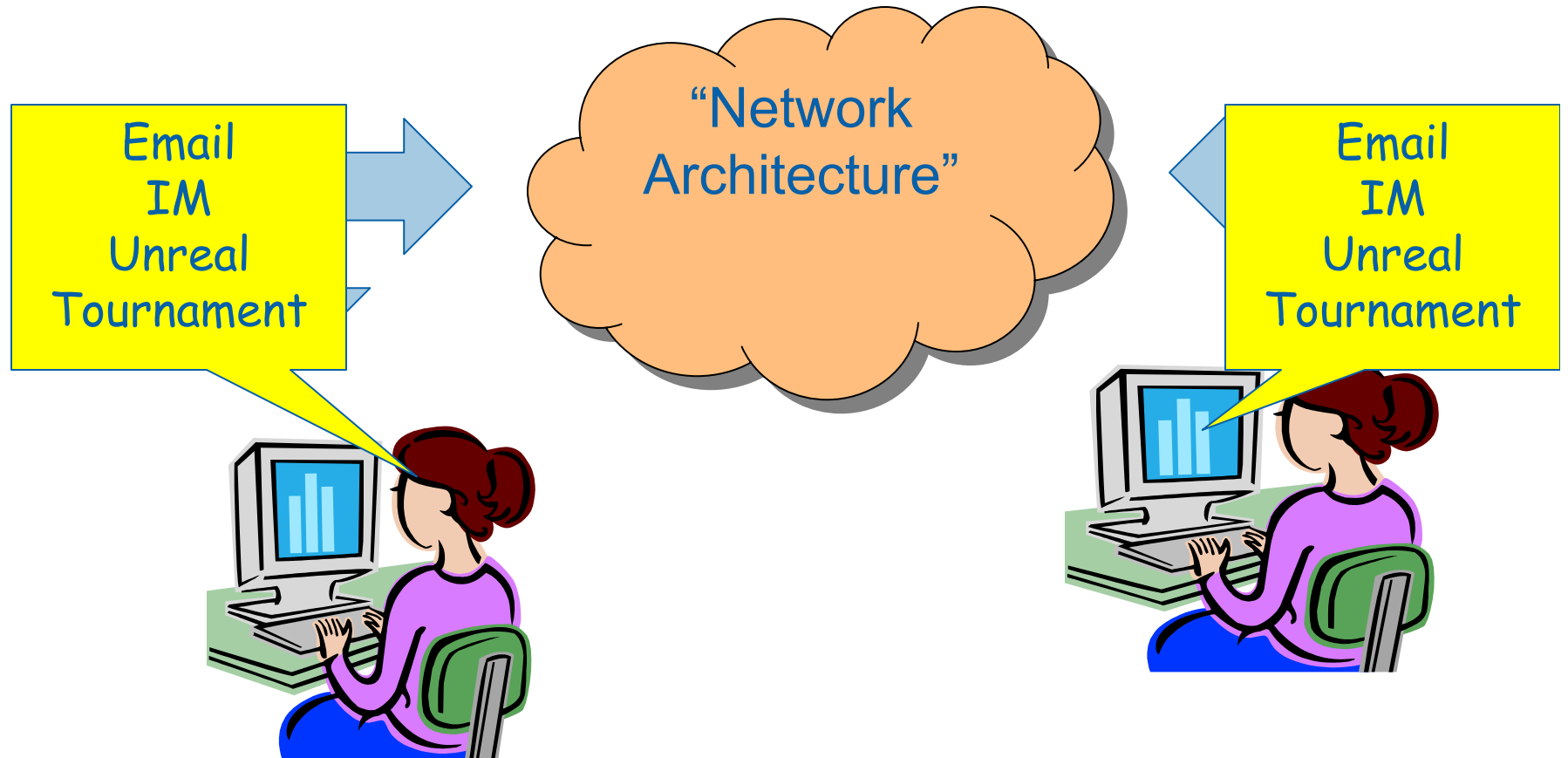
# What has the “Internet Architecture” done for us lately?

- Interoperability:
  - No – not really 😊
- Uniform API:
  - Bad thing: hides useful features of the underlying network.
  - c.f. cross layer optimisations and other oxymorons
- Provider framework:
  - Has any tier-1 ISP ever made significant profit from offering IP service?
  - Net Neutrality Debate etc etc

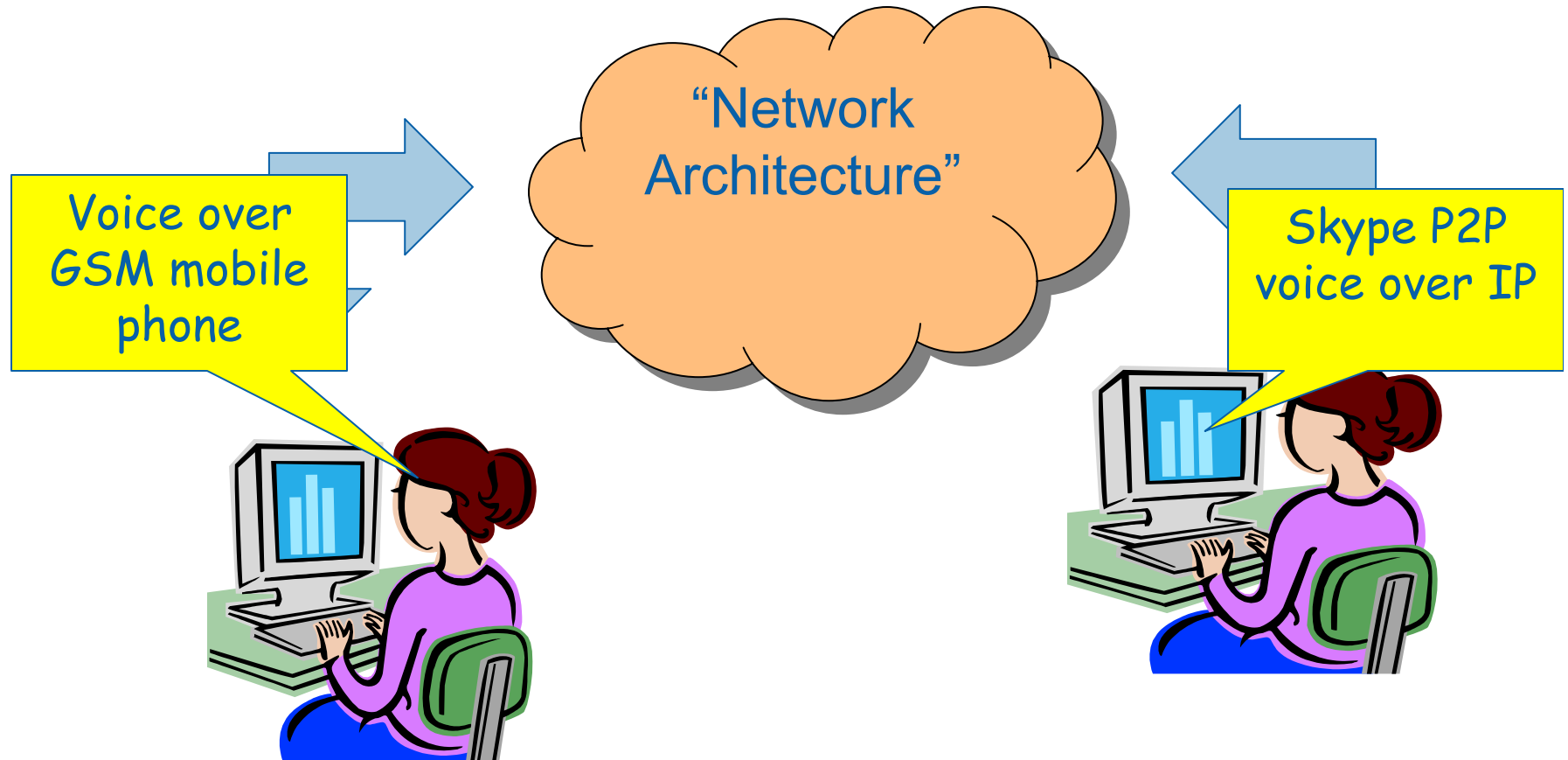
# What has the “Internet Architecture” done for us lately?

- It *used to* enable rapid innovation
- Claim: lack of attention to value flow & economics was a Good Thing!
  - High commercial value blunts innovation (c.f. other industries)
  - Disruption is bad for business
- Idea: goal of networking research should be to enable surprising things

# One (purist?) view of network architecture

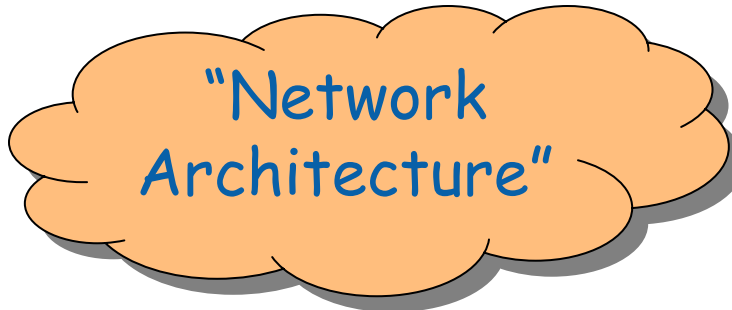


# One (pluralist) view of network architecture



# Different (separatist/corrosive/layered) view of network architecture

"Systems people"



"Networking people"

# A third way...post internet arch ("post" can bind to net or arch:)

- Sidestep the whole purist/pluralist debate
- Avoid an architecture completely
  - It's unnecessary
  - It doesn't address any problems
  - It may be counterproductive
- Let me give a detailed worked example from the Huggle Project :- IST-4-027918-IP

# The alternative

- There is no architecture
  - Just a bunch of computers with links
  - Removes “semantic bottleneck” from application writers
- Embrace “radically heterogeneous networking”
  - Two end-systems should be able to communicate even if they have *nothing* in common: protocols, address realms, semantics.

# Surely something must be common?

- Of course: but it is only *the application*.
  - ⇒ that what is common is that which is application-specific
- Even DTN has single overlay architecture
  - (bundle routing)

# By way of example: Hagggle (== DTN bis): A framework for Networking Mobile Users

- Wireline Networks: Solid
- MANET: Liquid
- Hagggle: Gas
- Manage Phase Changes between these – reality of wireless is that we need to move from
  - Gas -> Solid
  - Gas -> Liquid
- More often than we previously realized

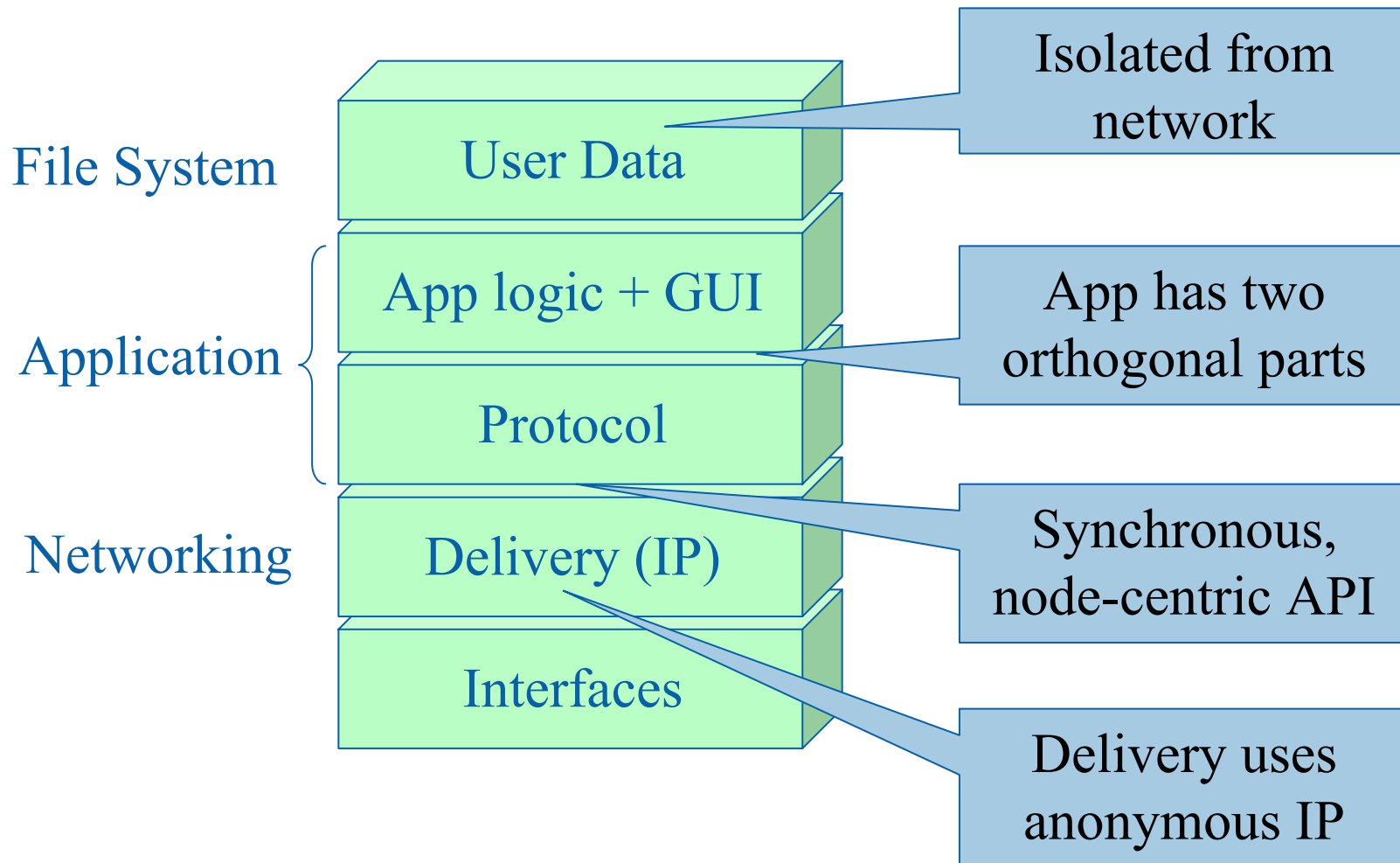
# Our approach: Hagggle

- Clean slate design of networking subsystem for mobile devices
- Hagggle: a layerless, but modular network architecture
  - designed around the needs of mobile users
- Aims:
  - Allow applications to take advantage of all types of data transfer (neighborhood, infrastructure, mobility)
  - Allow networking endpoints to be specified by user-level naming schemes rather than node-specific network addresses,
  - Allow limited resources to be used efficiently by mobile devices, taking into account user-level priorities for tasks

# Design principles for Hagggle (not an architecture, no no no:-)

- A. Forward using application layer information
  - A. De-layer
- B. Asynchronous operation
  - A. Do not assume path
- C. Empower intermediate nodes
  - A. Do not separate host from router
- D. All user data kept network-visible, and req/resp
  - A. Tx==rx==store - all equal class functions
- E. Exploit all data transfer methods
  - A. Greed is good
- F. Take advantage of brief connection opportunities
  - A. Impatience is good
- G. Empowered and informed resource management
  - A. Know and tell what it costs

# Current device software framework

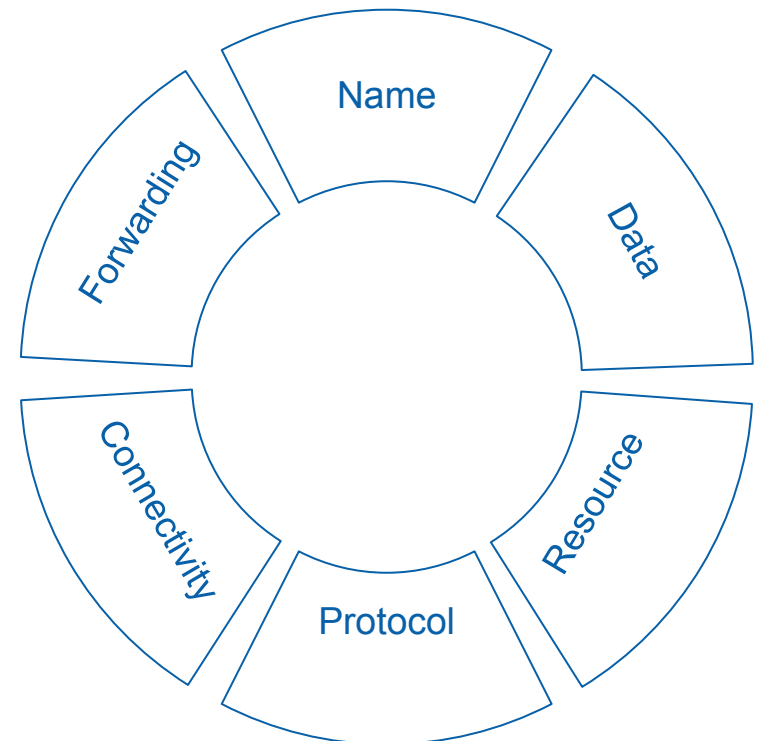


# Haggle Overview

- Clean-slate redesign of mobile node
- Spans MAC to application layers (inclusive), but is itself layerless – uses six “managers”
- API is asynchronous and data-centric
- Store-and-forward architecture with data persisting inside Haggle not separate file sys
- App-layer protocols (SMTP, HTTP, etc) moved into Haggle rather than apps themselves
- Naming/addressing done in graphs instead of stacks
- Resource management built-in

Applications (messaging, web, etc)

Haggle Application Interface



Connectivities (WiFi, BT, GPRS, etc)

# Data Objects (DOs)

- DO = set of attributes = {type, value} pairs
  - Exposing metadata facilitates search
- Can link to other DOs
  - To structure data that should be kept together
  - To allow apps to categorise/organise
- Apps/Haggle managers can “claim” DOs to assert ownership

## Message

DO-Type	Data
Content-Type	message/rfc822
From	James Scott
To	Richard Gass
Subject	Check this photo out!
Body	[text]

## Attachment

DO-Type	Data
Content-Type	image/jpeg
Keywords	Sunset, London
Creation time	05/06/06 2015 GMT
Data	[binary]

# Protocols

- Protocols are methods for communicating with other nodes, including Huggle-specific protocols as well as legacy protocols (e.g. email)
- Example: protocol to send an ADU over an ad-hoc WiFi connection given a neighbour's MAC address
- Example: protocol using keywords as addresses via Google – e.g. satisfy a query for “world news”
- Legacy example: encapsulate an ADU in an email and send it using SMTP to an email address

# Delivery uses user-level names(ANA)

- Plutarch, IPNL, RSIP etc etc etc
- Names come from:
  - Network, e.g. using neighbour discovery (12:AB:23:98:BE:FF)
  - Applications, e.g. Jon Crowcroft » jon.crowcroft@cl.cam.ac.uk
- Some names are also “addresses” i.e. data can be delivered to that name using one of the protocols available
- Delivery engine needs to:
  - Sense “nearby” addresses (e.g. Bluetooth inquiry gives MAC addresses, Internet connectivity means all email addresses are deliverable)
  - *Known-sender*: Map between ADU’s destination name(s) and addresses of suitable next-hop nodes
  - *Known-recipient*: Determine suitable nearby nodes which may be sources or help find sources for requested data
  - Describe these potential transfers and their benefits to the resource manager

# Resource management using tasks

- Task is a network activity
- Resource management uses a list of tasks mainly provided by delivery engine
  - E.g. perform discovery on interface I
  - E.g. send ADU X to neighbour Y on interface Z
- Each task has an associated benefit and cost
  - Benefit is specified by task provider. May be time-dependent (i.e. using a pointer-to-function)
  - To get cost, resource use is estimated, and then the "cost" is a function of the resource use \* the resource scarcity
  - Resource manager then schedules execution of tasks in order of highest benefit/cost ratio.

# So where are we?

- Problem definition and some idea of solution
- NOT architecture:- framework for others to use(Sourceforge)
  - Others develop use cases
  - Architecture will emerge from common uses
- Other work we've done:
  - Measurement/analysis of human connectivity patterns
  - Measurement of in-motion wireless networking properties
- Ongoing work in European project
  - More measurement and analysis of possible data paths
  - Implementation of modular software framework
  - Addressing challenges of naming/addressing, forwarding, security/trust, usability, *legacy compatibility*

# Back to FIND/GENI & Architecture/Testbeds & EU v. US

- As per my contentious (\*) abstract:
  - FIND and GENI are to fill a funding gap US has
  - EU doesn't have gap
  - Architecture research is philosophy, not science or engineering
- The devil is in the details, so:
  - Let 1000 architectures bloom
- World will pick a winner
  - Some ego will declare it the new architecture and become
  - The *mother-in-law* of the Next Generation Internet
- FP7 should fund good communications systems research as usual.

