



i n v e n t

# How to Build a Global Services Network Without Really Trying\*

Rick McGeer,  
June 19, 2007



\*Or, why design is a really **bad** idea

# Changing the world...

- A great distributed services network can change the world
- But to change the world, it has to be adopted
- We all want to change the world, but we all want to design a diamond
- Nobody adopts a diamond
- What we need is a ball of mud.

APL is like a diamond. It has a beautiful crystal structure; all of its parts are related in a uniform and elegant way. But if you try to extend this structure in any way - even by adding another diamond - you get an ugly kludge. LISP, on the other hand, is like a ball of mud. You can add any amount of mud to it and it still looks like a ball of mud.

-- Joel Moses

# The Seven Essential Features of a global hypertext system (from Xanadu)

- Bidirectional Links
  - See what's linking to a document
- Fine-Grained
  - Link to any part of a document
- Extrinsic links
  - Anchors supplied by persons other than the original author
- Detectors
  - Alert readers when documents have changed
- Types/filterability/endorsements
  - Permit readers to filter documents
- Version control
- Scalable without brute force

# What the web had

- Fine-grained
- Extrinsic (kind of)

# Original web was minimal

- http protocol spec was less than five pages
- Original http server was nine lines of shell script
- But minimality was the key feature
  - Made web easy to adopt (anybody can install and maintain a nine-line shell script!)
  - Making a web page was easy
    - Look, ma, three tags! (actually 5 – obvious three + `<html>`  
`<body>`)
  - Real value was in *many* web servers, *many* pages
- Much more complicated now (but already adopted)

# What the Web has now

- Bidirectional Links (Google, sort of)
- Fine-Grained
- Extrinsic links (Google's find feature, kind of)
- Detectors (Wiki, RSS)
- Types/filterability/endorsements (Blogosphere)
- Version control (Wiki)
- Scalable without brute force (CDNs)
- Other things Xanadu didn't think of (executable services)

The most creative thing I do is the imperfect invention. A perfect invention is sterile; it inspires nothing. An imperfect invention inspires others to create something far better than any of us could have done individually

-- James Lovelock

# Moral of the Story

- You need less than you think
- Extensible systems can be grown after the fact
- If it's really needed, somebody else can put it in
- Features are the enemy
  - Every feature is a barrier to adoption
  - The only design principle is simplicity.

# Moral of the Story (II)

- Society-scale information systems are *grown*, not designed.
- Growth medium is adoption and human ingenuity
- We need to think about systems in a biological and social context
  - Growth agents: incentives to adopt/features which spur adoption
  - Toxins: Barriers to adoption and maintenance
- **Every feature which doesn't have unanimous demand is a poison.**
- **Every page of a spec is a disease**

# So, What Are we Trying to Build?

- Service Layer: A network of *distributed services*
  - Critical, pervasive, robust, services
  - Programs that run *everywhere, all the time*
  - Robustness, low latency everywhere
- Network Layer: Allocatable, controllable network
  - High-QoS applications require support at the network layer
  - Guaranteed bandwidth at all time scales
  - Low latency

GENI's mission now sounds like "how do we take PlanetLab to the next level?" ....PlanetLab is going to be a hard act to follow.

-- John Ousterhout

# What Are the Problems with PlanetLab?

- Oversubscribed/undermaintained nodes
  - Nodes are oversubscribed
  - No incentive to provide resources
  - No incentive to maintain nodes
- Management is centralized
  - We will hit the limit in the very near future
  - Already experiments in federation
- No support at layer 3 or below (overlays only)
- Fix these three things and we can go home (and it will cost a *lot* less than \$350 million)

# Small Problem (Layer 3)

- Need routers which permit end-system control of bandwidth and latency
- New generation of routers and network protocols which do just that
  - TIA-1039 Explicit Rate Flow Management Protocol (ERFM)
  - SAFIRE (George Mason University)
  - Anagran FSR-1 (Anagran Networks)
- Others sure to emerge over the next few years

# Major problem

- Incentive to maintain/support nodes
- PlanetLab largely a research vehicle to date
  - Few end-user services
  - EUS's restricted to content distribution, multicast (largely unused)
  - Key: CDNs make existing services BetterCheaperFaster, don't offer new functionality
- Learning
  - Sysadmins don't care about/provision stuff that isn't operational
- Need to make PlanetLab an **operational** resource at hosted institutions
  - Need a popular end-user service which depends on PlanetLab

# One idea...Croquet

- Open-source virtual-worlds based collaboration system
  - Based on *replicated computation*
- Keys
  - Minimal latency (< 100-150 ms to all participants in a session)
  - Persistence of actions
- Needs
  - Network of 24x7 super-peers to host sessions
  - Network of 24x7 discovery servers
- Many research issues
- PlanetLab could become the server/discovery backbone.
- Sure to be other services

# Croquet, Today and Tomorrow

- “Mosaic” of the Virtual Worlds Simulation Based Collaboration Internet
  - Buggy (but we’ll fix that)
  - Distributed without “NCSA ‘What’s New’ Page”
  - Would the Web have taken off if Mosaic had been distributed this way?
- *Desperately* needs super-peer session host network
  - Default place to go when Croquet is taken out of the box
- Question: how would I design a backbone for something like this?
  - Many interesting research questions (how do I exploit replicated computation to make distribution, update easy?)
  - How many active users/super-peer?
  - Can I use a multicast tree? What does this do to latency?
- Pick this service (or another you prefer) and design around **that...**



i n v e n t