Writing Network-Adaptive Applications

Armando Fox, Daedalus/GloMop

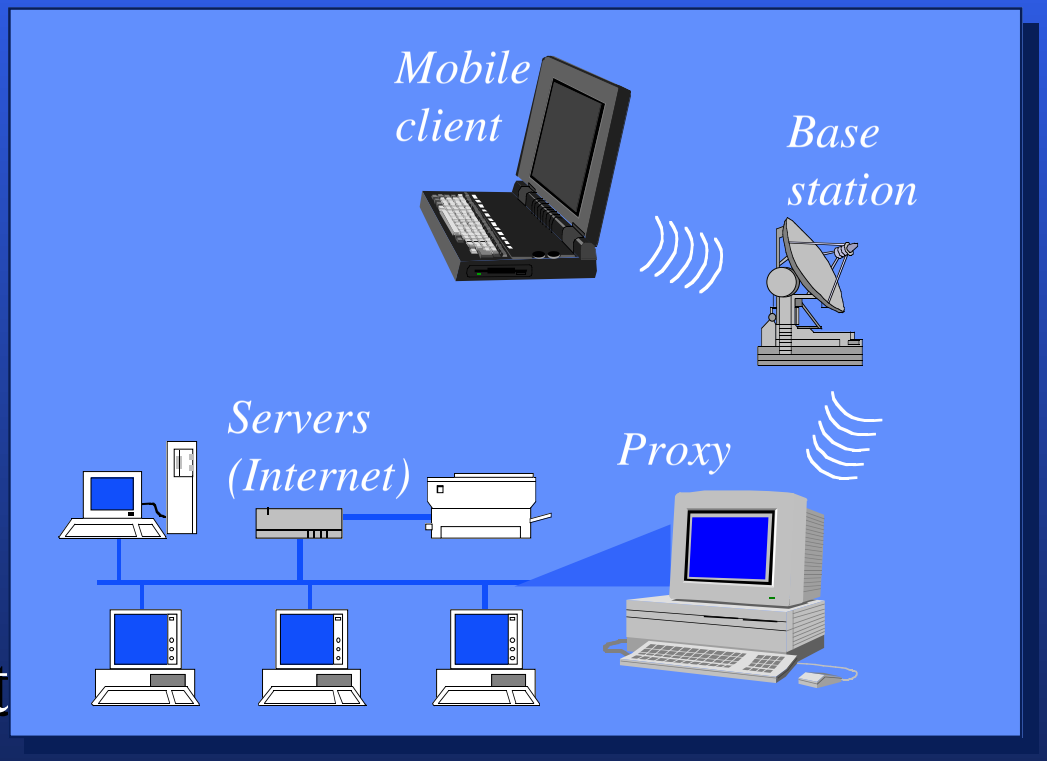
glomop@full-sail.cs.berkeley.edu

Overview

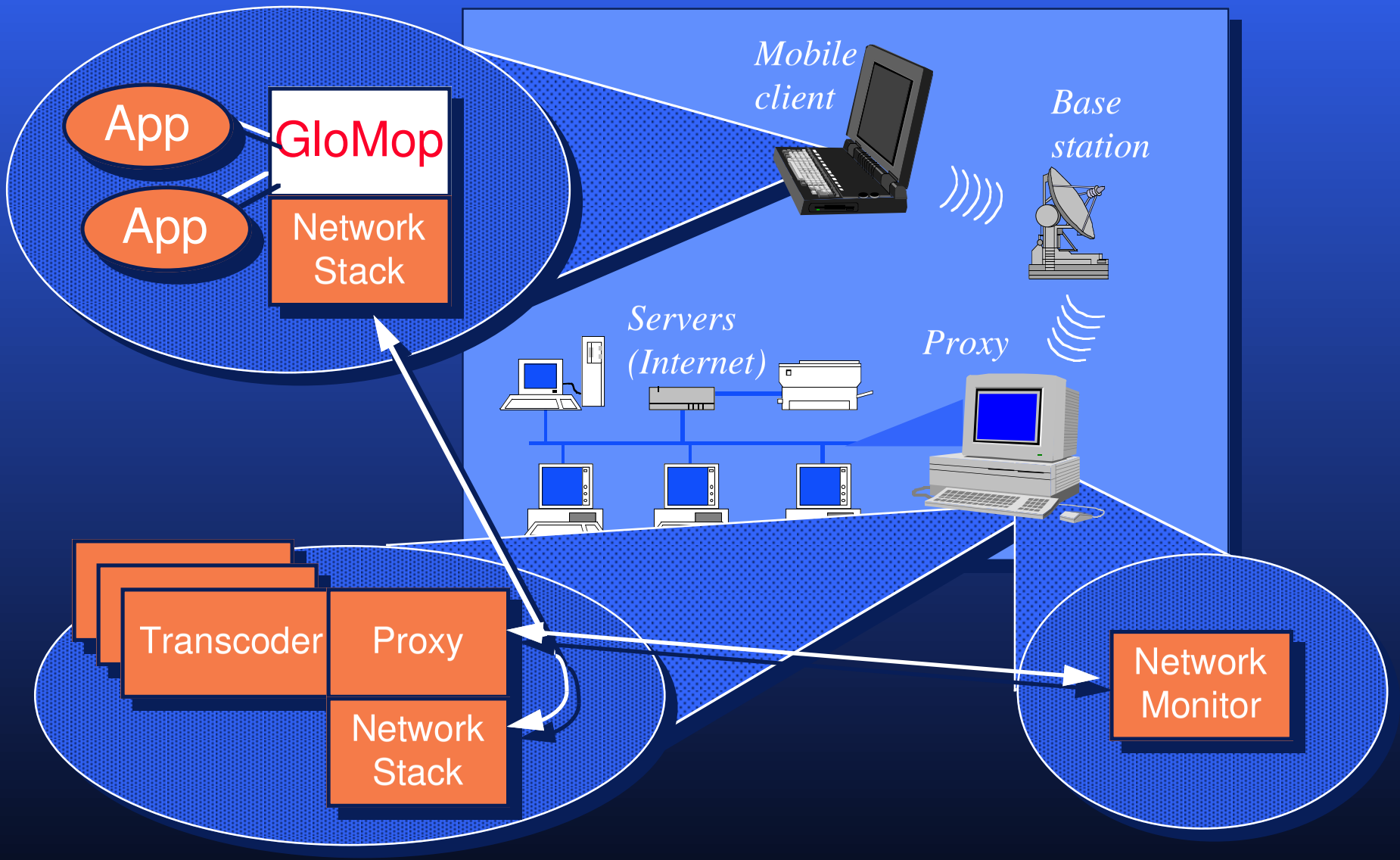
- Goals
  - Application's view of the proxy architecture
  - Real-time document distillation using a proxy
  - Adaptive response to network changes
  - How far can we go with HTTP?
- Non-Goals
  - Details of proxy-side implementation
  - Mechanics of network layer
  - Mechanics of handoff
  - ...things the application doesn't deal with directly.

Review: Proxy Architecture

- Proxy is at logical boundary of well-connectedness
- Trade cycles at proxy for wireless bandwidth



You Are Here


App
App
GloMop
Network Stack
Mobile client
Base station
Servers (Internet)
Proxy
Transcoder
Proxy
Network Stack
Network Monitor

Outline

- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

Document Structure: Chunks

- 4 *text/html* chunks (or 1 big one)
- 2 large *image/jpeg* (distillation probably needed)
- 3 small *image/gif* (distillation probably not needed)
- Document = collection of chunks

Programming Model

- Mobile clients want to *exchange documents* with fixed hosts
  - WWW surfing
  - Email
  - Groupware (calendar, document markup, etc.)
- But, mobiles differ qualitatively from fixed hosts
  - *Poor connectivity!* (Kbits/s, sometimes high latency)
  - Less horsepower, smaller screen, less memory...
- Use datatype-specific *distillation* to match document representation to mobile constraints
  - Each chunk contains only one datatype
  - Chunk is the basic unit of data

Subtype-Specific Modules

- Parameterized real-time transcoding between representations of some data type
  - Example: *image/gif* → *image/pict*
  - Example: *application/postscript* → *text/html*
- Proxy uses statistical models of transcoding latency & compression to satisfy QoS constraints
- Transcoding may occur at proxy or client
  - Prefer proxy, where cycles & memory are cheaper
  - But...may push to client to get a more efficient representation for transmission

QoS: High-Level View

- User-specified document transmission constraints
  - Latency bound: "at most *k* seconds"
  - Quality bound: "quality at least *k*"
  - Cost bound: "at most *k* dollars"
  - Power bound: "aggressively batch transactions"
- Defaults supplied by template hierarchy:

Document → *Application* → *Subtype* → *System*

Outline

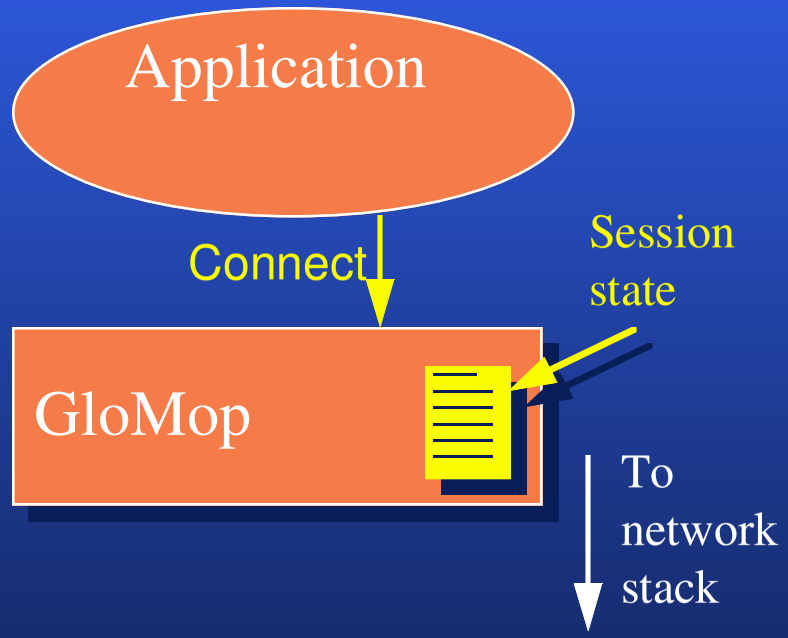
- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

Scenario: MagicLink WWW

- 4-gray screen (*not color*)
- 480x320 pixels (*not 1024x768*)
- 2400 or 14.4 modem (*Not 10Mbit Ethernet. Not even ISDN.)*
- Screen contrast is considerably worse than this picture suggests

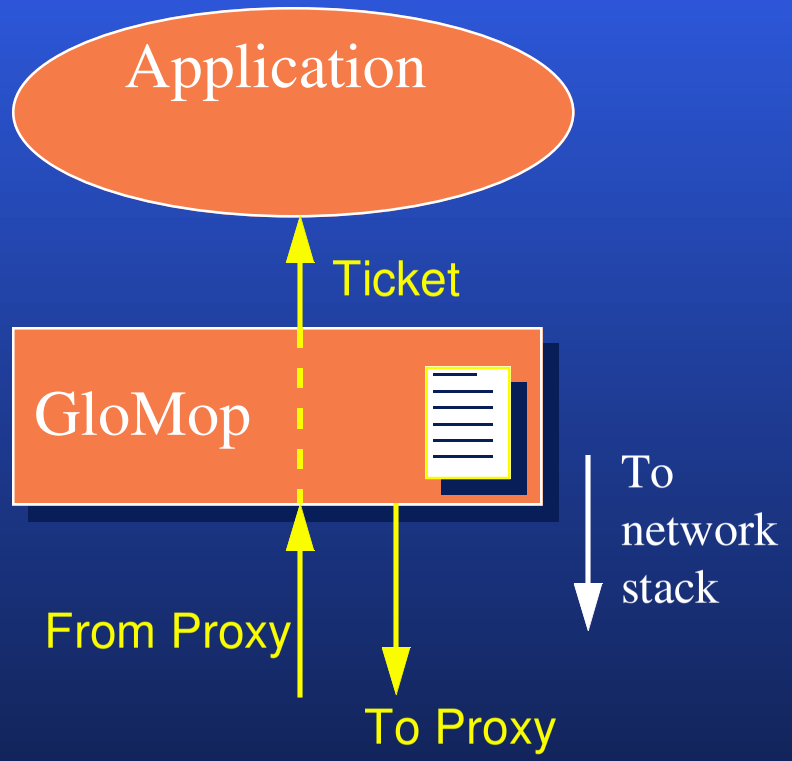
Connect to Proxy

- Connect & authenticate
- Type/Subtype registration
  - which subtypes client can easily render



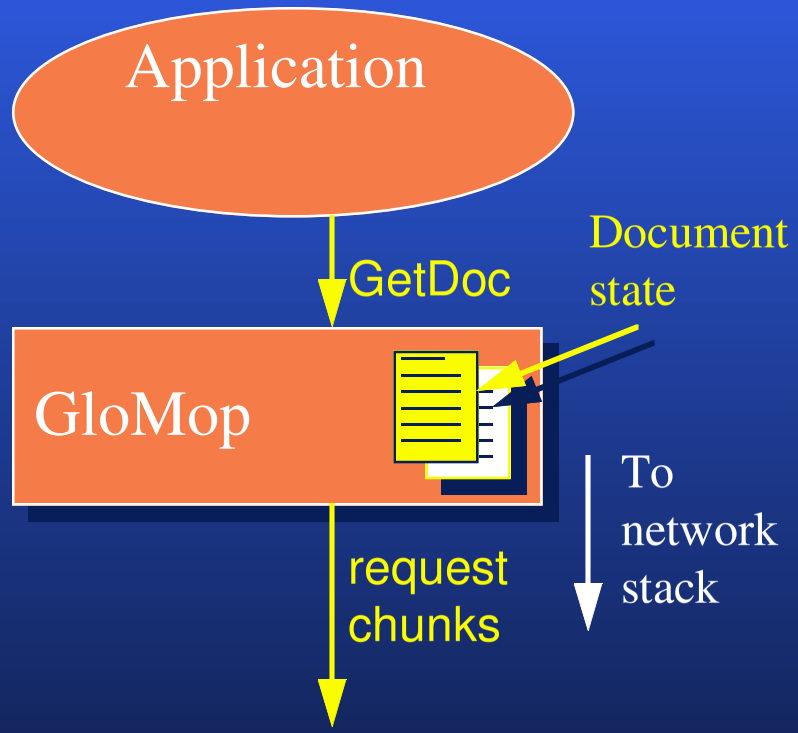
Connect to Proxy

- Connect & authenticate
- Type/Subtype registration
  - which subtypes client can easily render



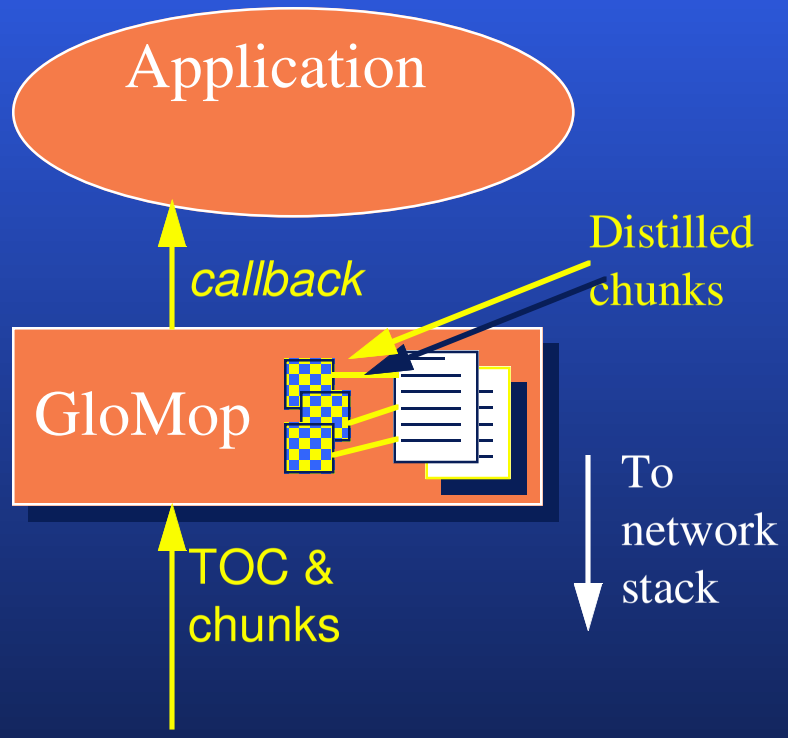
Request Page

- Connect & authenticate
- Type/Subtype registration
  - which subtypes client can easily render
- Request Document
  - Document name (URN)
  - Quality of Service prefs
  - Maximum preload count



Request Page

- Connect & authenticate
- Type/Subtype registration
  - which subtypes client can easily render
- Request Document
  - Document name (URN)
  - Quality of Service prefs
  - Maximum preload count
- Callback when chunks arrive



Refining a Distilled Image

- Ask proxy for *refinement* of an inline
  - New QoS prefs
  - Refinement axes
- Proxy already has original
- Creates new version on the fly
  - Larger
  - Only the subregion specified by user (refinement axes)

Review: Retrieving Document

- GetDocument(*docLocator, qosPrefs, preloadCount, callbackProc*): asynchronous
- GetChunks(*docID, whichChunks, qosPrefs, callbackProc*): asynchronous
- Refinement of a chunk: use *docLocator* embedded in chunk; modify *qosPrefs.refineAxes*
- Refine axes semantics are datatype-specific
- Also a linear "gross quality scale" that maps [0,*maxQ*] to points in refinement space

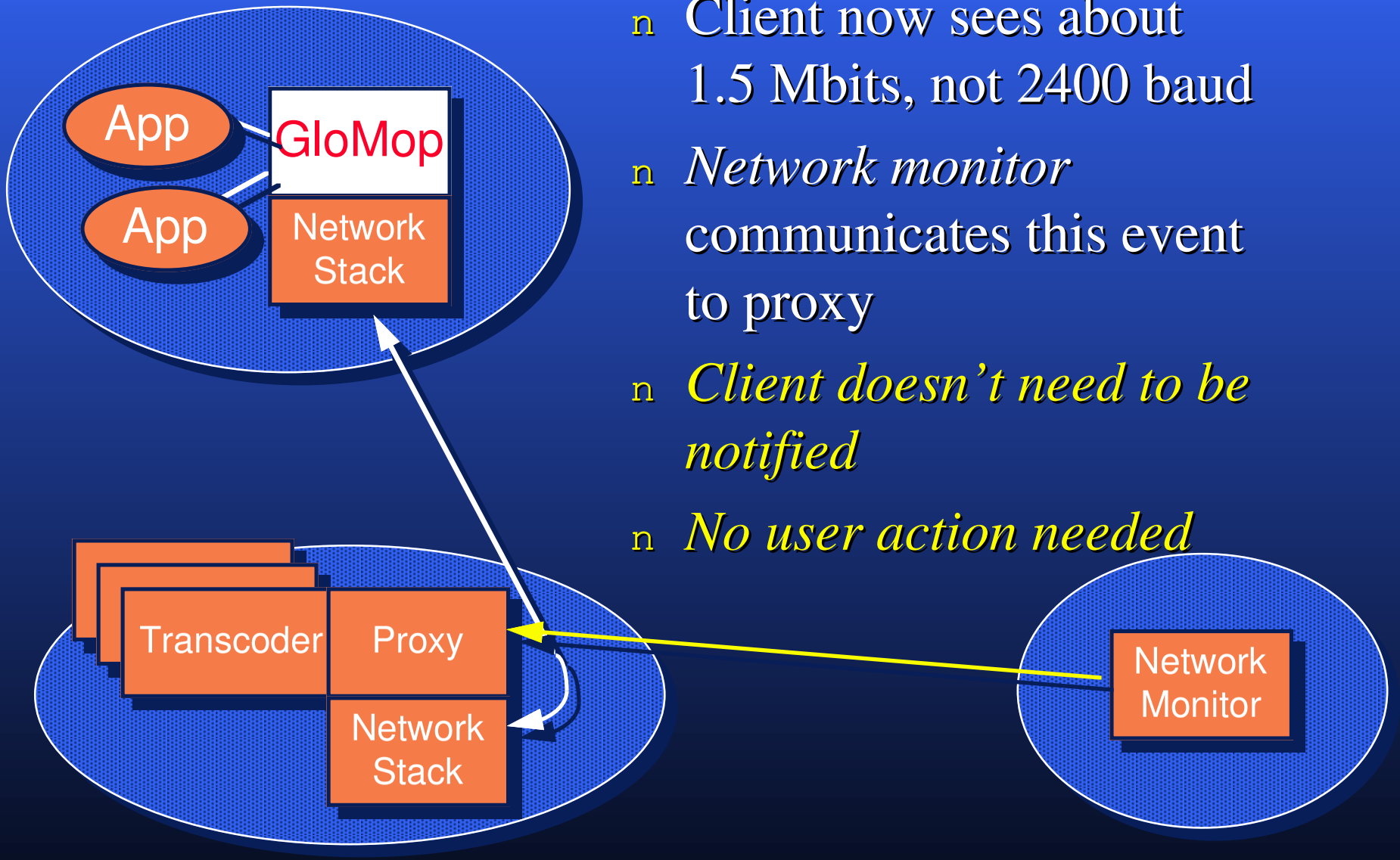
Outline

- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

Building Adaptive Applications

- Change in network conditions
  - Enter/leave radio shadow
  - Horizontal or vertical handoff
- Proxy will be notified first, via Network Monitor
- Application *may* be notified, if some parameter leaves a previously-specified range

User Powers Up WaveLAN



- Client now sees about 1.5 Mbits, not 2400 baud
- *Network monitor* communicates this event to proxy
- *Client doesn't need to be notified*
- *No user action needed*

When Is User Action Needed?

- Sudden disconnection while bandwidth-intensive operation in progress
- Excessive bandwidth loss, latency increase, dollar cost increase, etc. may require user action.
- User defines “excessive”
- *Out-of-band signalling* from proxy to GloMop causes callback into application

Review: Adaptive Behavior

- "Noncritical" network event will cause proxy to automatically adapt distillation behavior
- RequestNotify(*eventParams, callbackProc*): defines critical boundary values for network parameters
- Out of band notification to application when parameters leave this range

Outline

- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

Current Status

- Starting POSIX/BSD implementation of GloMop (client-side middleware)
  - Tcl/Tk and TkPerl interfaces for app development
  - BSD/OS on ThinkPads w/multiple wireless NI's
- Starting Java implementation of GloMop
  - Cross-platform portability
  - Executable content makes it easy to dynamically add SSM's and services
- Development on Magic Cap & Newton
- Summer: Windows 95?

WWW Anytime, Anywhere

- Hurry or we'll miss the bandwagon.
- Why no browsers for MagicLink & Newton?
  - Modems are 2400 baud!
  - Screens suck
  - *NewtonWWW* shareware: uses *gif2pbm* and a (non-adaptive) server running on a WS
- Bandwidth & display constraints can be addressed by proxy
  - Pythia HTTP proxy already does this
  - Leverage *NewtonWWW* to start (HTTP over serial?)

Conclusions

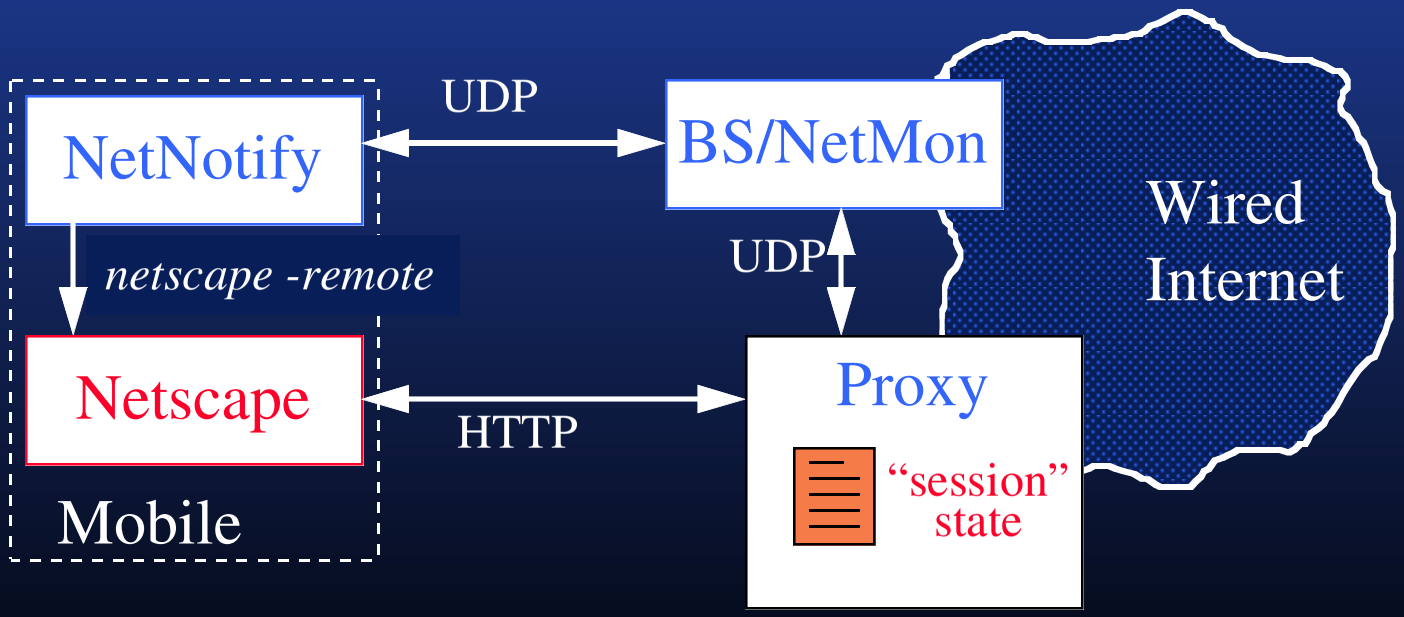
- Distillation & refinement by proxy are the foundations of bandwidth management
- Dynamic distillation decisions enable adaptive applications Æ exploit wireless overlay concept
- 6 months: “WWW anytime, anywhere”

Outline

- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

The Role of HTTP

- Transport protocol from GloMop to proxy
- Allows backward-compatible with stock Netscape (e.g.)
- Keep *all* state at proxy, explicitly register & deregister a "session"



Open Issues: How Far Can We Leverage Existing Infrastructure?

- *Massive Netscape/Windows infrastructure*
- Simple, established protocol: HTTP
- *Massive Netscape/Windows infrastructure*
- Uniform formats for all requests
- *Massive Netscape/Windows infrastructure*
- Can avoid writing brand-new apps
- *Massive Netscape/Windows infrastructure*

...you get the idea

Outline

- Elements of programming model
- Distillation and refinement mechanisms
- Adaptive behavior mechanisms
- Six-Month Plan
- Brainstorming: Potential Role of HTTP
- Discussion & feedback

Discussion: API

- Reality check: in line with KISS?
- Streams support: doesn't fit document model
- Writing middleware for PDA's
  - "Not your father's Unix"
  - Better than writing "TCP Lite" stack?
- Multiple Logical NI's (really a Proxy issue)
  - Demultiplexing (bandwidth striping)?
  - Map QoS's to choice of NI?
  - Who turns NI's on and off? (user? app? proxy?)—power management implications

Limitations of Existing Infrastructure

- *All* state must be kept at proxy
- Can't hide/recover from sudden disconnection
- Hard to build stable statistical model of network
- upload scheduling must be done in network layer!
- Can't preload stuff (use spare BW to hide latency)
- Can't add new services (fax back, text-to-speech email, etc.)
- TCP-based, ASCII protocol