GloMop Update

Armando Fox & Steve Gribble

glomop@full-sail.cs.berkeley.edu
GloMop Update

- GloMop Vision & Design Requirements
- What’s New
- Progress on Open Issues
- Research Directions
- Current Work: Building & Measuring
- Research Issues Summary
Vision: Proxy Benefits

- Mantra: *Access Is the Killer App*
- Address client hardware limitations
- Address client software/protocol limitations
- Dynamically react to changing network conditions by adapting content automatically *(on-demand dynamic transcoding)*

GloMop: High-Level Abstraction

- **Document structure abstraction**
  - Uniform document structure made of *chunks*
  - Each chunk is a single MIME type
  - Type chosen to fit client’s ability to render

- **Session-level network abstraction**
  - Proxy is a logical connection
  - Network scheduling invisible to applications
  - Asynchronous, multithreaded communication

- **Distillation abstraction**
  - User preferences for distillation
Requirements

- Security and authentication
- Support existing applications & infrastructure to promote widespread adoption
- Scale to very large numbers of users
- Uniform architecture from laptops to personal communicators
- Sensible interface and implementation for distillation/refinement preferences
What’s New: Research Infrastructure

- **gmwish**: GloMop Client API
  - Tcl/Tk shell for writing GloMop-aware apps
  - Connection to proxy, asynchronous callbacks, document request, chunk management…

- **Test-driving GloMop**: “image browser” app
  - Debug user preferences mechanism
  - Debug refinement interface

- **Charon**: lightweight indirect authentication via Kerberos IV
  (A. Fox and S. Gribble, submitted to MobiCom 96)
What’s New, continued

- **gmproxyd**: Modular Proxy
  - “Pluggable” client protocol filters
  - Remote controlled distillers
  - Pythia will be (re)implemented as a special case

- Better size prediction for GIF munching

- Proxy-Transcoder Manager (PTM) prototype
  - Load balancing of distillers on a NOW
  - Prototype implementation balances simulated loads
  - Integration into Pythia and gmproxyd: end of summer
#1: Support for Existing Infrastructure

- **Proxy-side Protocol Converters**
  - Pluggable proxy module
  - Translates existing protocols and formats to/from GloMop API calls
  - *Lose benefit of optimized single-connection protocol*
Alternative: Proxy’s Evil Twin

- Protocol converter *at the client*
  - Get benefit of optimized single-connection protocol
  - E.g. MOWGLI project (Univ. of Helsinki/Nokia)
  - *Inappropriate for “small” device*
Composing with Other Projects

- E.g. Harvest (distributed caching), Rover (queued RPC object model)
- GloMop protocol lightweight enough to carry “any” kind of traffic
- Proxy-side protocol handlers modular and extensible
#2: Interesting Range of Clients

- PDA’s, midrange cel phones, pagers, personal communicators, next-generation InfoPad
- Full GloMop is too heavyweight: devices too small, and there are way many of them
- **GloMop Lite**: minimal app support layer/set of abstractions for these clients
- What low-cost hardware support makes sense? (encryption, DSP, etc.)
#3: Scalability—Computation

- WWW traffic bursty at all levels: long idles, very high peaks (Crovella & Bestavros, SIGMETRICS 96)
- Document sizes: heavy-tailed, with small documents accessed much more frequently
- Detailed statistical profiling of WWW underway (Gribble, Gauthier et al.)
- Caching works well at all levels (Various)
Scalability: Load Characterization

- Playback of UCB CS HTTP logs
- One HP 715/80 PA-RISC workstation
- Distiller performance based on recent measurements

![Latency vs Number of Users Graph]

- 10 users
- 16 users
- 20 users
- 24 users

Distillation Latency
Remote control distillers
- Avoids proxy touching every byte of originals
- Proxy and distillers can communicate on physically separate fast network (e.g. Myrinet)
- Side effect: distributes “cache” of open documents
#4: Dynamic Adaptation

- Can automated statistical modeling determine optimal target format?
- How to specify hardware/software constraints?
- How to trade off distillation parameters?
- User interface for distillation prefs?
#5: Open Issue—Delivery Classes

- No support yet in `gmproxyd`
- How many distinct classes, and which ones?
- Interaction with link-level CBQ?
- Doesn’t make sense for GloMop Lite…

We need to get the prototype fully working in order to explore this.
Projects in Progress

*Need to build & observe system to “debug” various elements of it.*

- **gmwish** and image browser
  - user prefs, refinement, dynamic adaptation

- **Berkeley TCS, Geoworks Inc.**
  - characterization of heavy-load operation
  - load balancing on a NOW

- **Wink Communications**
  - Content adaptation for “really small” clients
  - Order-of-magnitude larger number of clients
Existing infrastructure: use protocol filters

Scalability: initial numbers encouraging, PTM prototype

Huge numbers of small clients: Nokia, Wink

Dynamic adaptation UI & computation: sample image browser

We need to build & deploy systems,
so that’s what we’re doing.
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

GloMop Update
#5: Distillation/Refinement UI

- Prototype image browser written with gmwish

- Pythia web proxy