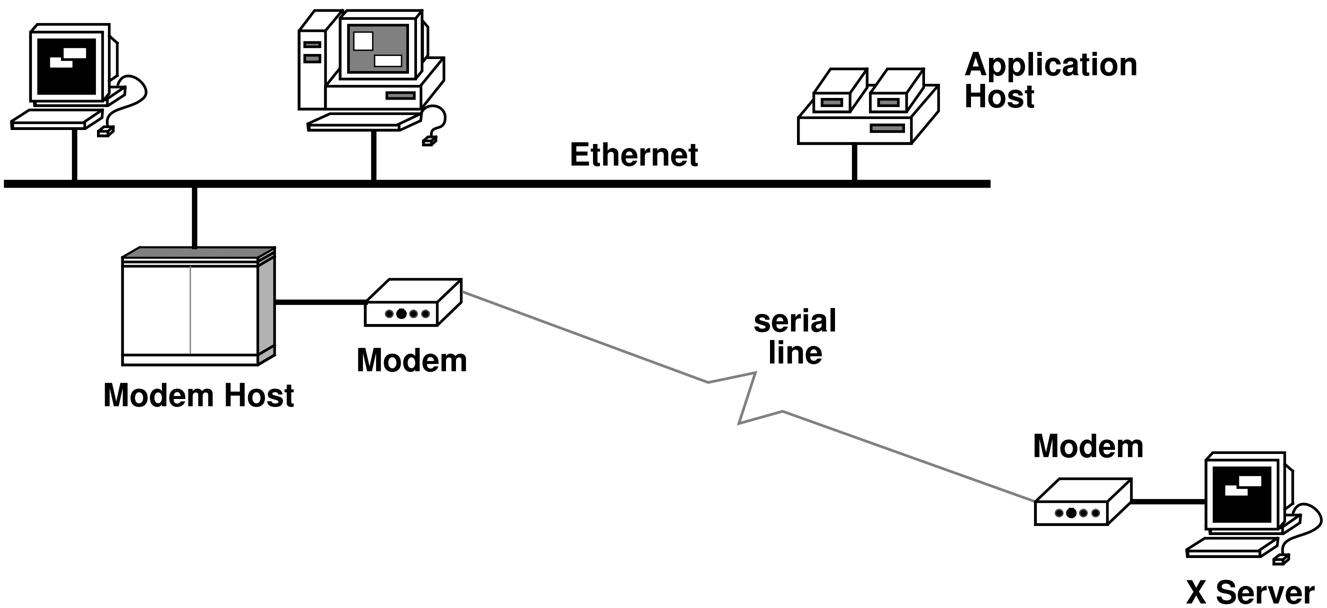
An Update on Low Bandwidth X:

A Standard for X and Serial Lines

Jim Fulton
Network Computing Devices

What’s the Problem?

- Need a standard for X and Serial Lines



Previous Approaches

- X server on host; escape sequences over serial line
  - Does not allow local clients (esp. window manager)
- Raw X over PPP or CSLIP (plain SLIP not usable)
  - Relies on compressing modems
  - Does not remove X protocol redundancies
  - Does not reduce serial port I/O
  - Trades latency for reduced CPU load
- Host-side pseudo-server, special wire protocol
  - Cache information on host side
  - Send deltas against prior packets when possible
  - Squeeze redundant data out of X protocol
  - Compression in or out of modems

LBX

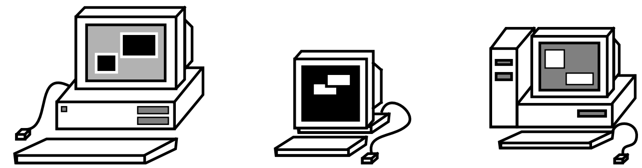
- Based on concepts in NCD's XRemote
  - Adds lots of host-side caching
  - Use PPP/CSLIP as transport
  - Reencode packets to take less space
- Transport-independent
  - PPP/CSLIP over serial lines, packet radio, etc.
  - TCP/IP over wide-area networks (WANs)
  - DDCMP over serial lines
- Uses existing network protocols when possible
  - File and Font Services
  - Print Services for local printing
  - Telnet (for terminal emulation)

Goals

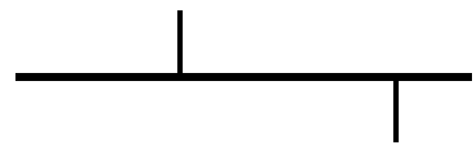
- Transparent to applications; binary compatible
- Standard for R6
- Be "reasonably" faster compared to predecessors
- Support at least 1 serial session at a time
- Scalable performance
  - Minimal functionality on small hosts and X servers
  - Extra functionality on more powerful systems

Target Platforms

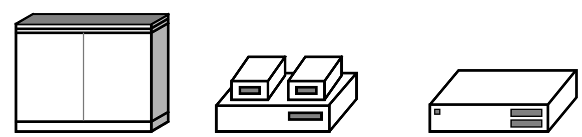
- Displays
  - pcs, terminals, workstations



- Connections
  - 9600-baud modems
  - direct serial lines
  - packet radio
  - wide-area networks
  - sample implementation TCP over PPP/CSLIP

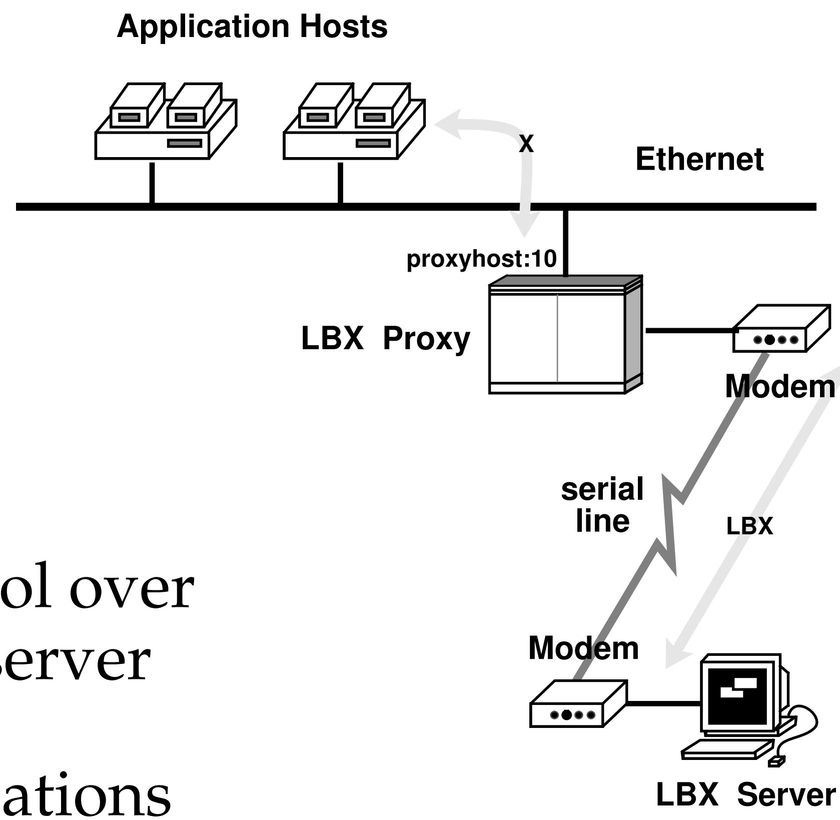


- Proxy Hosts
  - RISC workstations or servers
  - PCs and older CISC minis
  - terminal servers



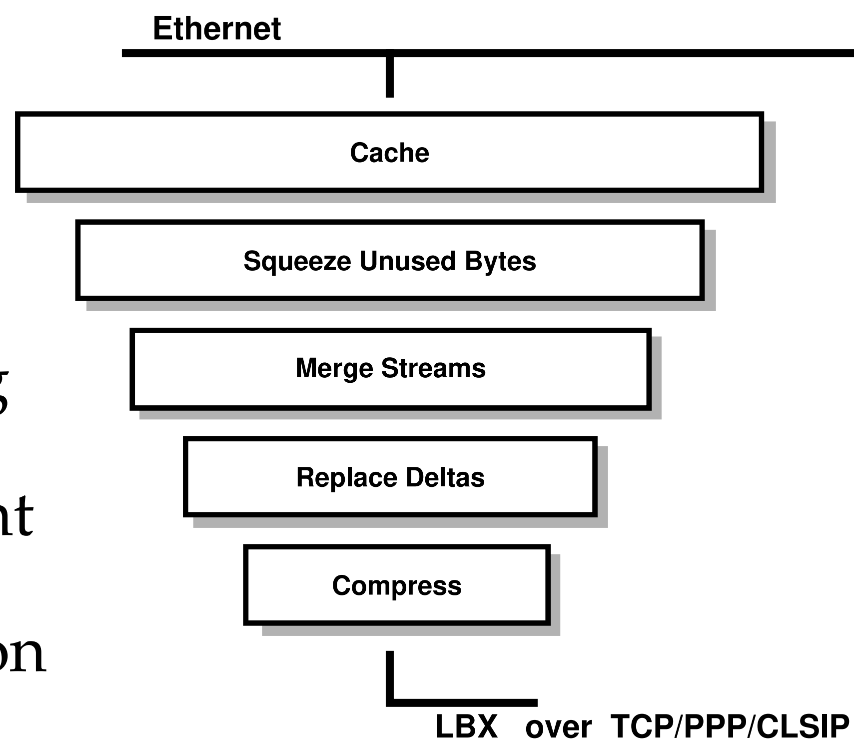
LBX Architecture Model

- Client/Server
- X applications talk to Proxy pseudo-server
- Proxy talks LBX protocol over serial line to LBX in X server
- No changes to X applications



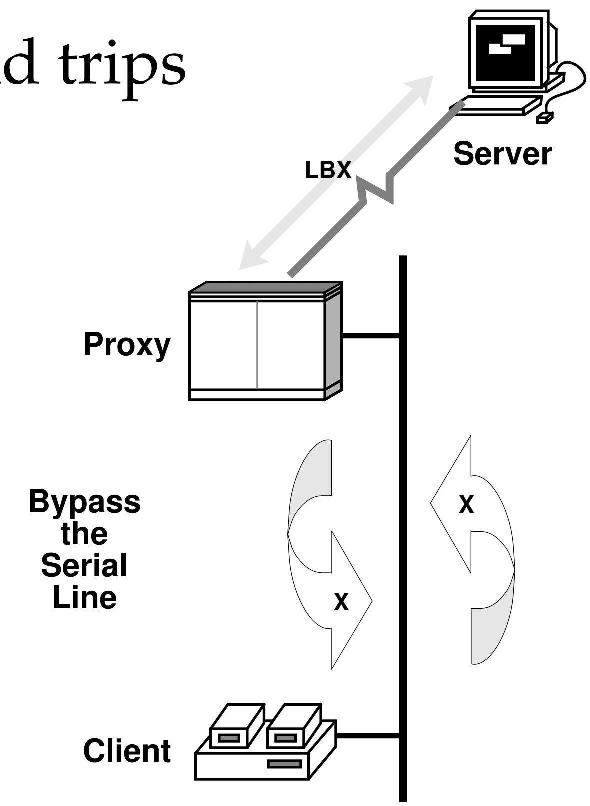
Major LBX Functional Blocks

- Lots of caching
- Better packet encoding
- More delta replacement
- Negotiable compression
- Use existing transports

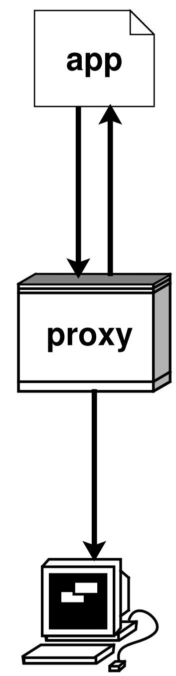


Caching

- Save data in proxy to avoid round trips
  - Application start-up
  - Interactive user input
- Keep large data to avoid waste
  - Connection setup information
  - Big properties
  - Font metrics
  - Keyboard mappings
- Other relatively static data
  - Atom values
  - Color cells and names



Connection Setup



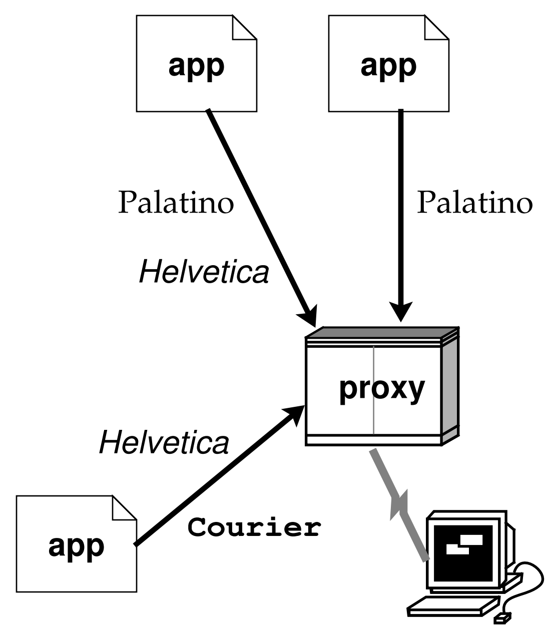
- Client initiates connection to proxy
  - But, thinks it is talking to server
  - Sends authorization data
- Proxy decides whether or not to allow
  - May ask server for help
- If okay, sends X setup data to client
  - Data usually the same for most/all clients
  - Each to cache
  - Roughly 80-100 bytes
- Client then reads RESOURCE_MANAGER property

Properties

- RESOURCE_MANAGER for .Xdefaults
  - Read on every client startup
  - Often 1000s and even 10000s of bytes long
  - Rarely changes, so easy to cache
- Cut-and-Paste Selections
  - Frequently both apps on same side of proxy
  - Nice to avoid fetching back up over slow line
- Drag-and-Drop Selections
  - Rapid and frequent messages between source and destination
  - If on same side of proxy, nice to avoid slow line

Sharing Fonts

- Fonts often shared among applications
  - Common look and feel (menus, labels, text, symbols, etc.)
- Font Metrics are big
  - 12 bytes per character
  - Latin1 fonts are ~2,900 bytes
  - Kanji fonts are ~95,000 bytes
- Font data does not change
  - Makes fonts ideal to cache
  - Really want to avoid redundant data transmissions



Keyboard Mappings

- Keysym Table
  - Maps <keycode,shift/lock> to keysym
- Size of Table: 1600-3200 bytes
  - Row of keysyms for each keycode
  - 100-200 rows
  - 2-4 keysyms per row
  - 4 bytes per keysym
- Read by Every Client
  - At startup, or
  - On first key event

Tab			
grave	asciitilde		
F2			
F1			
Control_L			
Shift_L			
Escape			
Q			
1	exclam		
F3			
Alt_L	Meta_L		
comma	less	kana_ comma	kana_ NE
p	P	thorn	THORN

Atom Values

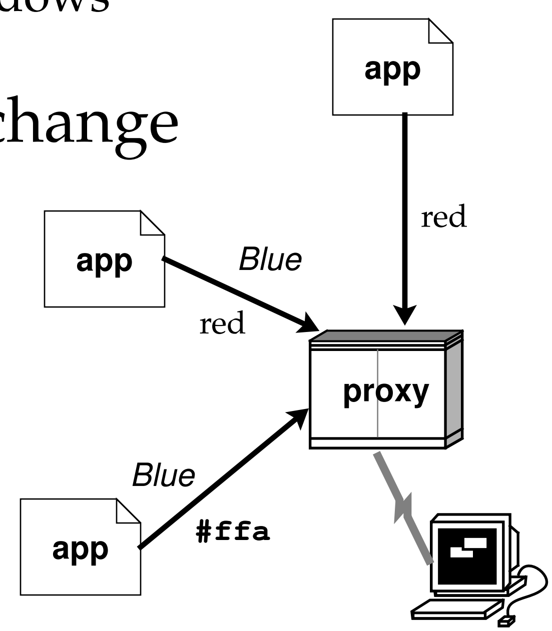
- Used to represent strings
  - Window Management (WM_*, _MOTIF_*, _OL_*)
  - Selection Names
  - Font Properties
- Heavily used by GUIs at startup
  - ICCCM window properties
- Unchanging and undeletable
  - Easy to cache

Sharing Color Names and Pixel Values

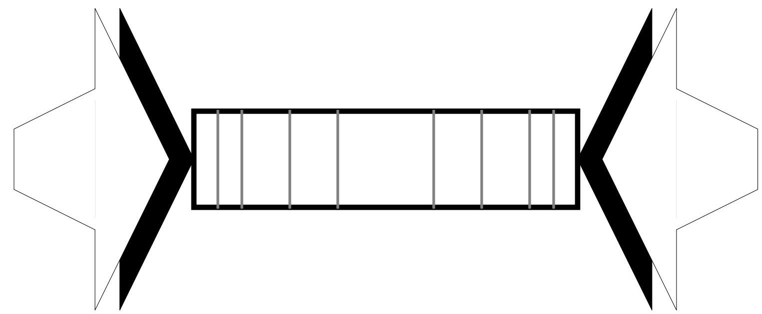
- Colors typically shared among applications
  - Common look and feel
  - Often allocated at startup
  - Also allocated when needed for new windows

- RGB values of color names don't change

- Read-only pixels don't change while allocated
  - Can cache known pixels
  - Avoids round trip when user is waiting

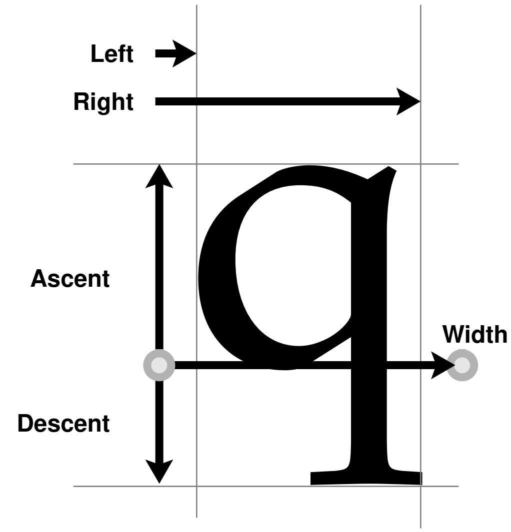


Better Packet Reencoding



- Image compression
  - CCITT Group 4 for bitmaps
  - PackBits for pixmaps?
- Font metrics
  - Normally 2900 bytes in Latin-1 and 95,000 in Kanji
  - Can squeeze down to ~1200 bytes for Latin-1 and 45,000 for Kanji
- Keyboard mapping
  - Normally 1600-3200 bytes, 50-70% of which contain 0
  - Represent as sparse array
- Events
  - Eliminate unused space

Tighter Font Metrics



- Character metrics 12 bytes per:
  - Left, right, top, bottom bearings
  - Logical width (distance to next character)
  - Undefined attribute flags
  - 2 bytes per field
- Most fonts can be squeezed down to 3-4 bytes per character:
  - Most bearings only need ¾ byte each (3 bytes total)
  - Terminal emulator fonts have constant width
  - Most attributes are always 0
- Similarly, cut headers from 280 bytes by 50%
  - Property atoms are almost half 0

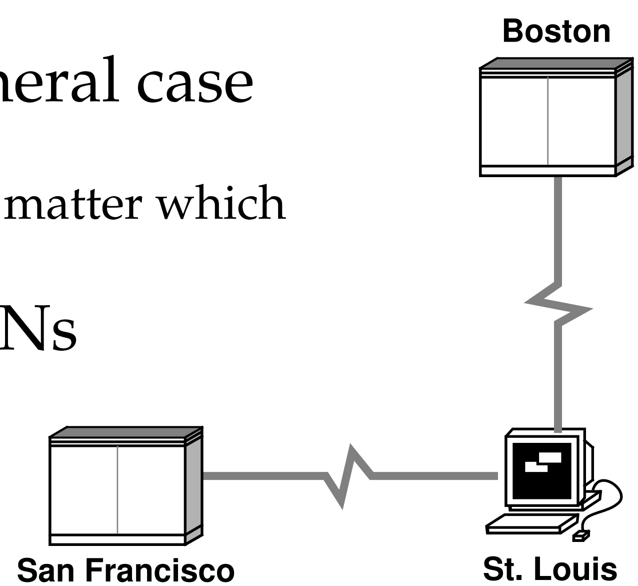
Squeezing Events

- Wasted space in lots of events
  - 15 bytes in Expose (out of 32)

MappingNotify	26	CirculateNotify	16
VisbilityNotify	24	CirculateRequest	16
Focus	23	Expose	15
NoExposure	22	PropertyNotify	15
DestroyNotify	21	GraphicsExposure	12
MapRequest	21	ReparentNotify	12
ResizeRequest	21	CreateNotify	10
MapNotify	20	SelectionNotify	9
UnmapNotify	20	ConfigureNotify	6
ColormapNotify	19	SelectionRequest	5
GravityNotify	17	ConfigureRequest	4
SelectionClear	17	Key and Button	1

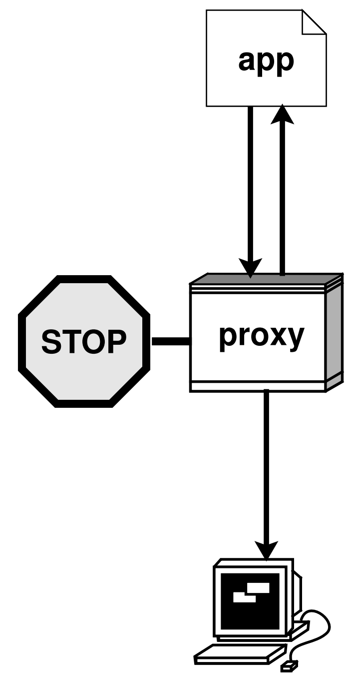
How Many Proxies?

- Just 1 is simplest
  - Single link to synchronize
  - Don't have to worry about sending info to other proxies
- Local clients probably force general case
  - Local clients are a requirement
  - If data can be in one proxy, shouldn't matter which
- Multiple proxies useful for WANs



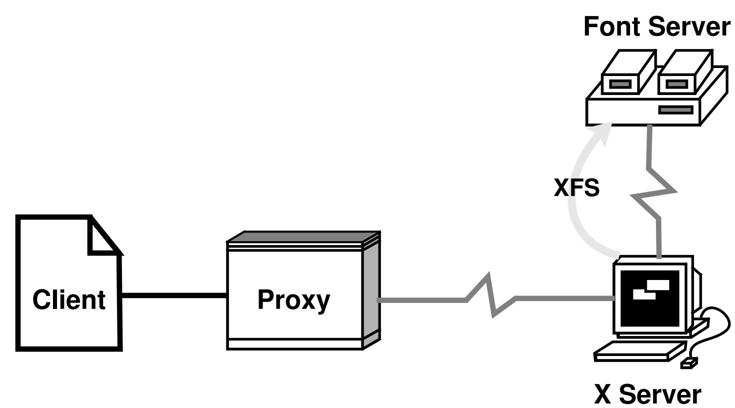
Authorization

- Server can delegate access control to proxy
    - Server notifies proxy when access lists change
    - Proxy decides whether or not to allow clients
    - Can ask server for help
- Proxy still passes info to server
    - Logging
    - Reuse with other services (*e.g.*, font service)
- Passing initial .Xauthority info useful
    - Duplicate what XDM does

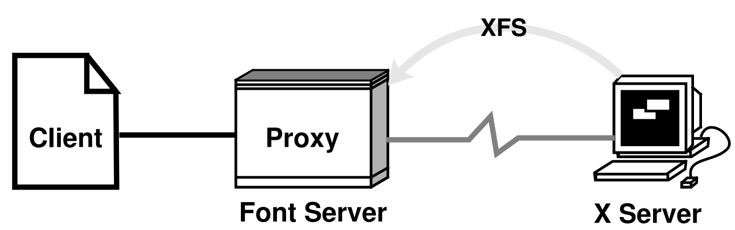


Font Access

- Direct
  - X server talks to Font Server
  - Easiest, cleanest method
  - Authorization problems



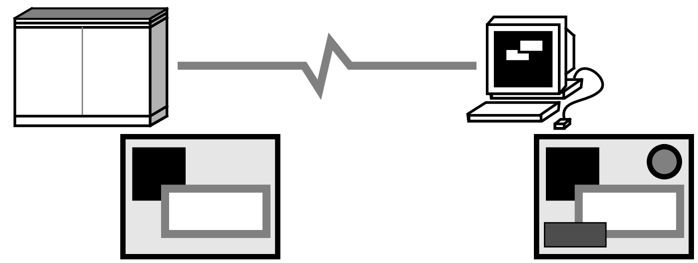
- Cheating:
  - Proxy can be a font server
  - Can reuse authorization data if necessary



Resource Allocation in Proxy

- Could avoid round trips if proxy handles:
  - Colormap entries
  - Atom database
  - Unmapped subwindows
- Server would pull ownership back when needed
  - Need to avoid thrashing, particularly if multiple proxies
- Proxy could update server when line not busy
  - Reverse is also true: proxy could load its caches “in background”

Shadowing the Window Hierarchy



- Could avoid round trips
  - QueryTree
  - GetWindowAttributes
  - GetGeometry
  - TranslateCoordinates
  - SendEvent
- But, synchronization with server difficult
  - Migrate ownership of hierarchy between proxy and server
- Local window manager complicates things a bit
  - Manipulates windows it doesn't own

Drag-and-Drop

- Most applications will probably be Dynamic
  - Lots of selection queries on every motion
  - Nobody really knows network impact yet
- Possible to handle in proxy
  - ChangeProperty, ConvertSelection, GetProperty, SendEvent
- Property caching in proxy
  - Avoids delays sending over wire uselessly
  - Issues with BadAlloc errors that would come if later sent to server

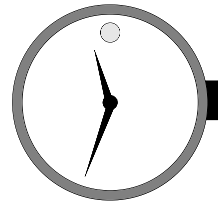
Motion Events

- Motion events used in important places
  - Key UI Gadgets (button boxes, menus, etc.)
  - Drag-and-drop
- Real-time motion can eat up a lot of bandwidth
  - 800 bytes/second
- Server can continuously update proxy
  - Recognize when proxy is likely to need motion events
  - Keep some number of events in flight (DEC SRC found 8 was good)
  - Reduce number of packets if line slows down

Grouped Events

- Some events often come in series
  - Exposures
  - ConfigureNotify
  - MappingNotify
- Proxy could "explode" special events
  - Server would send just one event
  - Proxy deliver to necessary clients
  - Proxy would split into multiple separate events if necessary
- Event reencoding and delta replacement does some

Status



- Paper in proceedings and on export.lcs.mit.edu as ~ftp/contrib/LBX-xconf93-paper.ps.Z
- Slides on export.lcs.mit.edu as ~ftp/contrib/LBX-slides.ps.Z
- Prospects good for meeting performance goals; protocol writeup in progress
- Specification for Public Review by July
- Code in R6 at end of year