Tech Talk
On security, networking, and performance
Dave Parker
UW SSEC
ADDE Security via SSH tunneling

- SSH fully encrypts all network traffic between the client and the server
  - Secure shell for remote access is the simplest case
  - SSH can do much more
- Tunneling sets up a local port through which all ADDE traffic is directed
ADDE Security via SSH tunneling

- SSH fully encrypts all network traffic between the client and the server
  - Secure shell for remote access is the simplest case
  - SSH can do much more

- Tunneling sets up a local port through which all ADDE traffic is directed
ADDE Security via SSH tunneling

- SSH fully encrypts all network traffic between the client and the server
  - Secure shell for remote access is the simplest case
  - SSH can do much more

- Tunneling sets up a local port through which all ADDE traffic is directed

Client

- ADDE
- Port 2345
- SSH

Server

- ADDE
- Port 112
- SSH

Port 22 (SSH)
ADDE Security via SSH tunneling

- User-configurable via DATALOC using SSHADD, SSHDEL, and SSHLIST keywords
  - DATALOC ADD TEST SLATE.SSEC.WISC.EDU
  - DATALOC SSHADD SLATE.SSEC.WISC.EDU davep /home/davep/.ssh/id_rsa

- Positional parameters are `user` and `private key` created with `ssh-keygen`

- Local port is randomly chosen

- Tunnels are shutdown automatically on exit
IPv6 support in McIDAS-X is non-trivial

MCTABLE.TXT stores IP addresses in dot-decimal notation (eg. 192.168.1.100)

Uses language library functions for conversion

ADDE header contains a single 4-byte Fortran integer signifying the IP address in network byte order
McIDAS-V and IPv6

- Java natively supports IPv6
- All non-ADDE transactions are IPv6-ready
- ADDE redesign is being discussed
  - Will take into account IPv6
  - Including backward-compatibility with ADDE 1 clients
McIDAS-V Performance Improvements

- **Image-by-reference**
  Image data is not copied when sent to the rendering device

- **Geometry-by-reference**
  Display geometry is not copied when sent to the rendering device

- **Non-Power-Of-Two (NPOT) texture sizes**
  Up to an order of magnitude improvement in rendering times and memory utilization when image textures do not fit nicely within power-of-two dimensions (eg. 1024x1024)

- **Grid Contouring**
NPOT

- Common on newer hardware
- Disabled by default in 1.2
  Certain uncommon hardware configurations crash
NPOT

- Common on newer hardware
- Disabled by default in 1.2
  Certain uncommon hardware configurations crash

POT only: Potential quadrupling of memory
  Eg. 1025x1025 → 2048x2048
NPOT

- Common on newer hardware
- Disabled by default in 1.2
- Certain uncommon hardware configurations crash

POT only: Potential quadrupling of memory

Eg. 1025x1025 → 2048x2048

- VisAD: Enable access to geometry by reference
- Enable access to image data by reference
- Enable Non-Power of Two (NPOT) textures
McIDAS-V Performance Improvements

**8-bit Color for Gray Scale**
- Detected automatically by renderer
- 2/3 reduction or 67% savings in memory
- Testing now, should be in 1.2r1

**Automatic tiling for large images**
- 4096x4096 is a common hardware limit
- Multiple textures used for GOES-E Vis (10500x10500)
- Example...
Texture Tiling
(10500x10500)
McIDAS-V Rendering Algorithm Improvements

- ByReference display update for fixed image dimension and coordinate system over a time sequence reduces display memory. Automatic fallback to default algorithm if it is not supported.

- Displaying in satellite data coordinates renders faster and uses less memory when the display coordinate system is the inverse of \((\text{line, element}) \rightarrow (\text{longitude, latitude})\). Allows space earth limb viewing from geostationary perspective.

- Reuse of display and color coordinates during rendering. If only the color table is modified, the display geometry is reused; if only a projection is modified, the color coordinates are reused.

- Capability to cache display images to disk.
Native Projection Limb
McIDAS-V Performance Improvements

Improvements to ADDE network performance

- Reduced ADDE requests from 5 to 2 when creating Imagery displays
  Load one frame from Unidata: 15s → 5s using 100mb less than 1.01

- Reduced ADDE request count by 50% when creating animations
  Load 20 frame loop: 151s → 75s using 250mb less than 1.01

Large loop performance

- Loading a 100 timestep loop of 1160x1265 MSG data took <10s to render, using 1.2gb after garbage collection
- Adding color enhancement takes 3s and an additional 500mb
Questions?