Hydra

A framework for web applications to access PSU mainframe data.

Mark Earnest
Senior Systems Programmer
Academic Services & Emerging Technologies
Pennsylvania State University
mxe20@psu.edu

PENNSTATE

The Problem

- Most PSU data resides on an IBM zSeries mainframe
  - The “master” copy of all student records and PSU financial data resides here
  - For security reasons, the mainframe does not run end-user accessible servers
  - Access to the data is only provided by 3270 terminal sessions
- In the early 1990's the World Wide Web came into popularity
  - There was a demand for easy access to mainframe data via the web
  - Students preferred to use the web to schedule classes, view grades, etc
  - A more “user friendly” interface for staff was desired for financial applications
- Security concerns increase exponentially
  - Migration from a centralized environment to a distributed environment
- There is a desire to retain legacy code
  - PSU invested money and programmer hours into developing the transaction handling code that runs on the mainframe

PENNSTATE

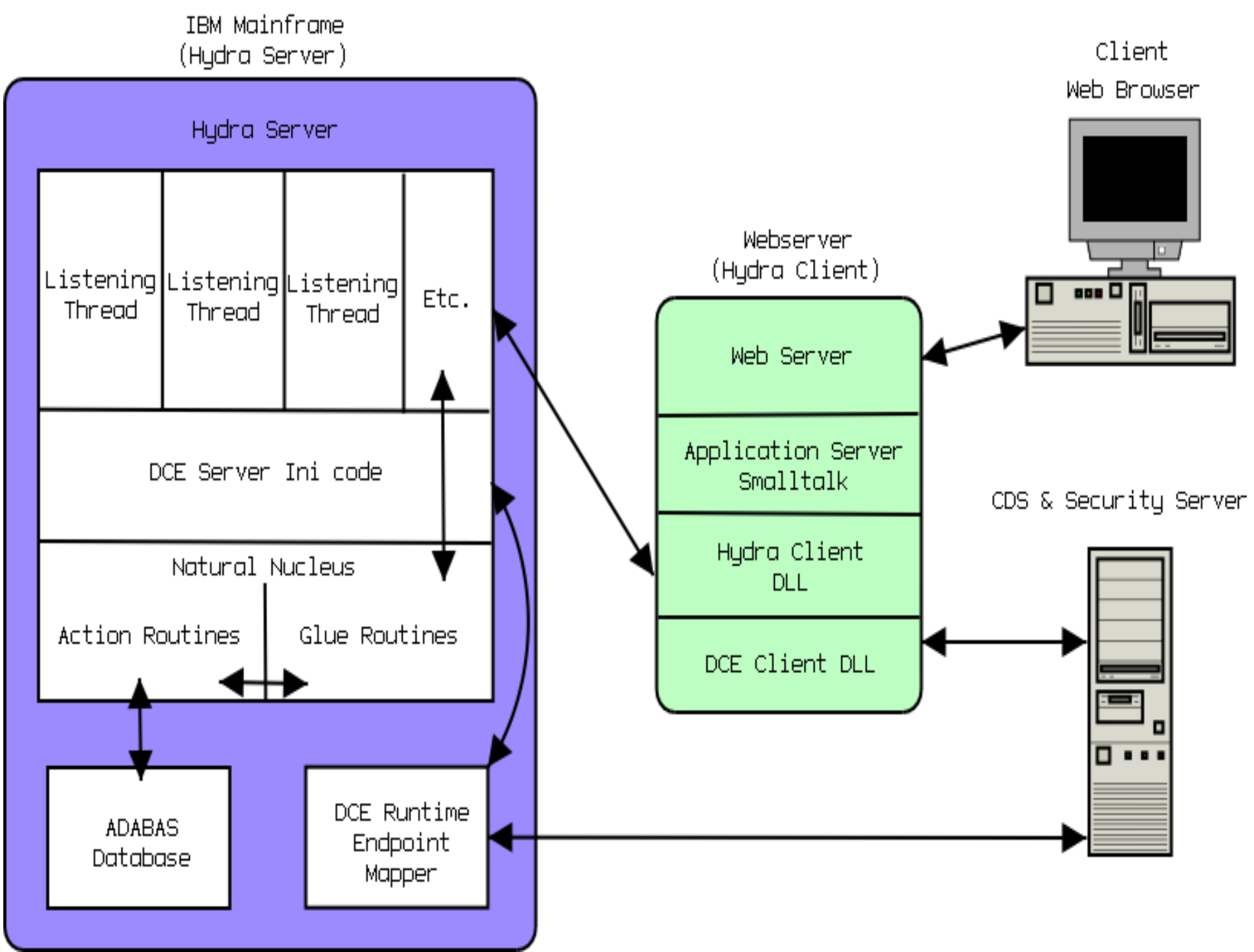
Our Environment

- System is an IBM zSeries (formerly s/390) Mainframe
- Two databases (student and financial) are SoftwareAG's ADABAS
- Transaction processing is handled using SoftwareAG's NATURAL (4th generation programming language)
- Online (3270 terminal) access to NATURAL programs is provided by SoftwareAG's COM-PLET program
- Access control is provided internally by Computer Associates's ACF2
- University-wide authentication/authorization and infrastructure provided by OSF's DCE

PENNSTATE

DCE Overview

- Security provided by MIT's Kerberos 5
  - Tickets extended to also store user attributes (groups)
  - Everything (users, server processes, client processes, etc) has a princpal
- Communication handled by DCE RPC
  - Provides several levels of security, including packet header and payload integrity checking as well as encryption (3DES)
  - Performance significantly better than SSL encrypted protocols
- Directory services provided by Cell Directory Service
  - Central data repository (ACL's on everything)
  - Acts as a “dynamic DNS” for DCE servers


IBM Mainframe
(Hydra Server)
Hydra Server
Listening Thread
Listening Thread
Listening Thread
Etc.
DCE Server Ini code
Natural Nucleus
Action Routines
Glue Routines
ADABAS Database
DCE Runtime Endpoint Mapper
Client
Web Browser
Webserver
(Hydra Client)
Web Server
Application Server Smalltalk
Hydra Client DLL
DCE Client DLL
CDS & Security Server

PENNSTATE

Hydra Server Startup

- Establishes Natural environment and connects to database
- Reads config file and sets global variables
- Starts logging
- Spins off Identity Management Thread
  - Uses keytable file to log into the dce cell and gets a login context
  - Refreshes context when necessary and routinely changes password
- Spins off Signal Handler Thread
- Registers itself in the local endpoint mapper and CDS directory
  - Endpoint mapper maps the port to the service name
  - CDS maps only the IP address and protocol type (TCP/UDP) to the service name
- Fires off listening threads and enters a wait state

PENNSTATE

CDS Naming

- Global namespace arranged in a directory tree structure
- DCE Servers have a specific entry in the CDS as well as a group they belong to
  - A server called DAS227SS1 has the CDS entry of /.../dce.psu.edu/subsys/servers/ais/hydra/prod/db227ss1
  - The server also belongs to the group /.../dce.psu.edu/subsys/servers/ais/hydra/prod/group/db227ss
- Servers can be called by clients either by actual name or group name
- Group names allow clients to call multiple, identical servers
- ACLs can be placed on CDS entries to limit which clients are allowed to call which servers

PENNSTATE

Hydra Client Startup

- Application calls MANinit()
- Reads configuration file, sets global variables
- Starts logging
- Spins off Identity Management Thread
- Requests list of partially bound binding handles from CDS
- Requests ports from endpoint mapper
- Spins off thread to store fully bound binding handle list
- Spins off thread to handle user login contexts
- Returns control to application, ready to accept further requests

PENNSTATE

Logging In

- Application calls MANLoginPW with a userid and password
- These are passed to a DCE function sec_login_validate_identity
  - Handles the Kerberos login
  - Builds context, a data structure containing authN/authZ info about the user
- Queries Bind Management Thread for a Hydra server handle
- A data structure called manhandle_t is built
  - Contains Kerberos authentication and DCE group information
  - Contains the Hydra server binding info for the server that is given to the user
  - Contains stat information about user (number of calls, mutex locks, etc.)
  - Data structure is stored in context management thread
- Returns address of manhandle_t to application

PENNSTATE

Calling Hydra (Client)

- Application calls MANSend passing the manhandle_t, natural application name, parameters, and empty array of pointers to point to return data
- Client binds to the server using the binding handle in the manhandle_t structure
- Client passes both its own and the user's authentication information to server for validation
- The actual RPC call is made, and return values are returned in a linked list
- Pointers to the data in the linked list are propagated into the return data array and returned to the calling application.

PENNSTATE

Calling Hydra (Server)

- A listening thread receives a bind request from the client
- The client's dce context and user's dce context are validated
- The natural program name is translated to a glue routine name
- The userid is translates to a student number
- The glue routine is called with the parameters passed from the client
- The glue routine makes calls to the action routines to get data from ADABAS
- The glue routine returns the data to the listening thread
- The listening thread packages data and returns it to the client

PENNSTATE

Server Lists

- By referring to groups of server instances, the clients get a list of available servers.
- This list refreshes every two minutes normally
- Each user is assigned a server by round robining through the list
- If a user fails to connect to a server (timeout) on a request:
  - The user is assigned a different server from the binding list and the request is automatically attempted again
  - The server is removed from the binding list so nobody else gets assigned to it
- If all servers are marked bad, the binding list will automatically attempt to refresh from the CDS
  - During this time, all requests will be queued until a valid server is found
  - If no server is found in 2 minutes, the queued requests will fail
- This all allows Hydra servers to be dynamically added, moved, or removed as required with no impact to the end user :)

PENNSTATE

Elion Today

- Normally two Hydra server processes running on the mainframe
  - Up to 6 during peak times such as grades week or add/drop week
  - Each server has 10 listening threads
  - Average request processing time < 1 second
  - We do not know what the max requests per server is… bottleneck is with the web servers :)
- Normally 4 Win2k Servers running IIS and VisualWave Smalltalk
  - Up to 8 during peak times (prior to Hydra, there were always 12)
  - Each machine can handle up to 600 connections with no performance degradation
    (prior to Hydra a web server would become unusable at around 70 connections)
- Additionally, grades and schedules are available on the portal
  - Running Sun Solaris
  - Portal application is written in c
  - portal.psu.edu

PENNSTATE

Hydra Benefits Wrap up

- Performance & Scale
  - Performance under heavy load improved
  - Number of webservers required vastly decreased
- Security
  - Data integrity checking & strong encryption
  - Integrates with PSU's central network security infrastructure
  - AuthN/AuthZ checking includes both client instances and end users
- Flexibility
  - Back end Hydra servers can be moved or removed without interruption of service
  - Servers can be added simply by bringing them up (no configuration changes)
  - Can run on most server platforms (win32/unix/linux)
  - Front end application can be written in any nearly language