Enterprise Control, Configuration,Logging/Alarming (using GNUstep-base)

by Richard Frith-Macdonald
from Brainstorm Mobile Solutions
(also maintainer of GNUstep-base)

History

- Mid 1980s: Brainstorm Computer Solutions distance learning using teletext software, Henley Management College, British Telecom
- Late 1980s: I joined, moving into data-mining software, and cellular phone (SMS) also with British Telecom
- Early 1990s: Distance learning moved to HTTP, data mining and SMS services expanded
- Mid 1990s onwards: SMS expansion, MMS, work with Vodafone and T-Mobile etc.

The problem - 24x7

- Control – we needed to be able to manage multiple processes on multiple hosts
- Configuration – we needed to be able to easily and reliably configure behavior of all processes
- Logging – we needed to be able to provide audit trails and diagnostic logs
- Alarming – we needed to be alerted when a problem occurs.

Context

- 1988 SNMP
- 1991 World Wide Web
- 1993 NeXTstep Intel released, GNUstep started
- 1994 OpenStep released, GNUstep named
- 1995 Brainstorm needed control/config/logging
- 1996 Postgresql, MySQL, SNMPv2
- 2000 Net-SNMP

Choices

(really easy at the time)

- Why not SNMP?
  - Expensive, closed source, incomplete, complicated
- Why not RDBMS?
  - Expensive, closed source, single point of failure, complicated
- Why GNUstep?
  - Distributed Objects, free software, simple!

Development

- Yes, this really is about 17 years old!
- Initial development was very rapid
  - The slowest bit was fixing memory leaks in the distributed objects system, which was really new when this started.
- Changes have been minimal
  - Bugfixes and gradual accumulation of minor feature over the years.
- Now cleaning up and simplifying for release as free software.

17 years later – why not SNMP?

- Now we have SNMPv2 and net-snmp is available … reliable free software
- But SNMP is still complicated and messy to code and to work with.
- It uses a database (MIB) which is hard to work with and doesn't handle complex data types well.
- Even if starting from scratch, I'd still not use SNMP.

17 years later – why not RDBMS?

- Now we have a reliable RDBMS in PostgresSQL (and MySQL is probably good enough too).
- But editing OpenStep style property lists to configure things (and add new config) is still easier/safer than hacking a DB using SQL.
- An RDBMS provides a smaller subset of the functionality then SNMP … so there's even more extra work to implement things on top of it.

The solution

- Control – a hierarchical architecture of communicating server processes based on common classes
- Configuration – a simple property list database sent out to the servers on a need-to-know basis
- Logging - Common provision of API, and support in the server process hierarchy at various points.
- Alarming – Common provision of API, and support in the server process hierarchy

Technologies

- Distributed Objects
  - NSConnection, NSPortNameServer
  - NSProxy, NSHost
- Property Lists
  - NSUserDefaults
  - NSString, NSDictionary, NSArray, NSData, NSDate
- Event driven
  - NSNotification, NSTimer, NSRunLoop

Control

- Hierarchical system
- Control server process (one per 'stack')
- Command server processes (one per host)
- CmdClient processes (many per host)
- Console processes (one per operator)
- Operator may start/stop/reconfigure any process while it's running.

Configuration

- Hierarchical
- Common information for whole stack
- Host specific information for single host
- Process specific information
- Multiple process instances, common config
- No local config required
- Dynamic update of running processes

Logging

- Multiple levels (Debug,Log,Warning,Error,Alert)
- Multiple logging destinations
- Configurable flushing conditions
- Debug fine control
- Automatic archiving of log files

Alarming

- Originally based on logging (Error and Alert logs)
- Rule based routing (source, type, regex)
- Multiple destinations (Email, SMS, Database)
- Coalescing/Buffering
- New Alarm system event start/end paradigm

Control server

- Provides a central point of control for a stack
- Loads configuration from Control.plist
- Controls and monitors Command servers
- Acts for and Reports to Consoles
- Processes any centralised logging/alarming
- Single point of failure?

Command server

- Provides a central point of control for a host
- Registers with Control server to get configuration
- Launches and shuts down Command clients
- Provides configuration to Command clients
- Forwards instructions and logs/information
- Single point of failure?

Command clients

- A Command client is any process which instantiates a library class (or subclass of the library class) to act as a client of a Command server.
- The instance registers with Command server to get config
- It accepts instructions from Command server
- It provides logging to Command server
- It provides API for other code in the process to do common stuff

Console process

- Command line process for operators
- Sends commands to any server process
- Supports manual startup/shutdown of servers
- Displays results of commands
- Displays logging output from server processes

Security

- A stack is inside a DMZ … security not an issue
- But … Console operator login uses username and password
- GNUstep Distributed Objects support authentication/encryption
- CmdClients can have a password configured
- So all inter-process communications can be authenticated and/or encrypted.

Installing a stack

- Install your GNUstep-base package (and start gdomap if it's not done automatically)
- Install your own software and Control.plist
- Start the Control server
- Start the Command server
- That's all … how can it be so simple?

Distributed Objects Nameserver

- The Control server registers itself with the name 'Control'
- The Command server registers itself with the name 'Command' and asks the DO system to connect to 'Control' on any host
- Each Command client asks the DO system to connect to 'Command' on the current host.
- Each Console process asks the DO system to connect to 'Control' on any host.
- The GNUstep DO system connects everything up for you.
- The NSUserDefaults system can be used to override the names and hosts used ... so you can control exactly what connects to what if you don't want to use a default architecture.

User Defaults System

- The standard NSUserDefaults system allows defaults to be set for each process, and then overridden at the command line
- The central configuration information from Control.plist is merged into the NSUserDefaults system for a Command client, but does not override command line settings.
- So you can control which Command server a client connects to by giving it a specific host name to connect to.
- You can access configuration information in your client code simply by using the familiar NSUserDefaults API.
- You can use the NSNotification system to observer changes in configuration (your client class can also override the method used to update defaults).

Property Lists

- Control.plist contains a dictionary of dictionaries with hierarchical configuration information.
- The key '*' is generic (all hosts) information, while the other keys are the names of hosts.
- Within a per-host dictionary, the key '*' is configuration for all processes, while other keys are process names.
- There is an include mechanism, to allow the configuration to be broken into multiple property lists (eg putting different hosts in different files).
- When a Console issues a 'config' command, the Control server checks for updates and pushes any relevasnt changes to all connected Command servers

Distributed Objects oneway void

- When a method is declared with a return type 'oneway void', the DO system knows it can send a message to a remote process without needing a response.
- This is key to most communications between the servers … we don't want one process to block waiting for another.
- This is used for a heartbeat mechanism between all processes … each end sends 'ping' methods to the other at intervals
- If pings are delayed too long, an Error is logged. If delayed even longer an Alert is logged and the connection is dropped.
- Instructions sent from the Console, and responses sent back to a Console are also asynchronous, as is most logging and pushes of configuration information.

What next?

- Preliminary 0.1 release
  - Refactoring and improving documentation
- Web based Console
  - Similar architectures
- Multiple stacks