Ice

CIS-3152, Spring 2012
Peter C. Chapin
What is Ice?

- Middleware system from ZeroC
  - Based on CORBA
  - Cross-Language
    - C++, Java, Python, Ruby, .NET (C#, etc), Objective-C, PHP
  - Cross-Platform
    - Windows, Linux, MacOS, Solaris
- Full featured
  - BUT... removes little used or “questionable” features from CORBA.
Slice

- [Add slides on the slice interface language]
Slice to Java Mapping

- [Add slides on the slice to Java mapping.]
- [Show examples: Chatter, FileServices]
Proxies

• Stringified
  • A way to represent proxies in human readable form.

• Direct vs Indirect
  • identity:tcp -h xyz.com -p 2000
  • identity@xyz

• Routed

• Replication
  • Proxies with multiple endpoints

• Replica Groups
  • Interaction with the Location Service
Servants vs Ice Objects

- **Ice Object**
  - An abstract concept of a remote object with methods.

- **Servants**
  - Servants incarnate “one or more” Ice objects.
  - Servants that incarnate multiple Ice objects
    - Get the identity of the object with each request
    - Useful when there are many, many Ice objects. For example, a database table.

- **Ice Objects are “virtual.” Servants are “actual.”**
“At Most Once”

- Ice guarantees...
  - A request executes once or not at all.
    - If a request does not execute, an exception is generated.
- Allows safe use of non-idempotent operations.
  - Idempotent operations are those where the effect is the same if they are executed more than once.
- Ice allows you to declare idempotent operations
  - In this case the Ice run time can provide more aggressive error recovery than normally possible.
Asynchronous Method Invocation

• By default Ice method calls are synchronous
  • Caller is blocked until method returns.
    – Could take a while even if the operation is quick due to network latency.
    – Upon return all results are available.

• Ice allows you to mark calls as asynchronous ("AMI")
  • Invoker passes "call back object"
  • Invocation returns at once.
  • Run time calls method on call back with result.
  • Servant unaware an asynchronous call was made.
Asynchronous Method Dispatch

- Server side analog to AMI
  - Servant informed of client invocation but uses its own thread to process it.
  - Thread in the Ice run time can now accept requests from other clients.
  - Servant informs local Ice run time when results are ready to be sent back to the client.
  - Servant thread can continue after data sent to client
    - Can perform clean up activities or other post processing.
Oneway Invocations

• Similar to AMI (asynchronous)
  • Call returns at once. Invocation done “later.”
  • Only allows data from client to server.
    – No information comes back... not even error information.
    – AMI allows return data via the callback object.
    – Oneway invocations are unreliable
      • Can't tell if they worked or not. Client just hopes.
    – “Best effort” semantics.
  • Server unaware that call was made “oneway.”
  • Can be batched to reduce overhead.
Datagram Invocations

- Call information transported using datagram protocol (e.g. UDP)
  - Similar to oneway...
    - Unreliable
    - Low overhead.
  - Additional errors possible
    - Completely lost invocations
    - Invocations might arrive in an unexpected order.
- Even less overhead than oneway.
- Supports multicast invocations.
- Can also be batched.
Exceptions

• Ice supports throwing exceptions over the network.
  • Two sources of exceptions:
    – Communication problems
      • Invocation never leaves machine
      • Target object does not exist or can't be incarnated
    – Ordinary failure of the called method
      • Requested operation could not be completed.
  • Exceptions due to communication problems reported via exception types in the Ice namespace.
  • Other exceptions are defined by the user as usual.
Services

- Higher level features built on top of the low level system. The following services ship with Ice:
  - Freeze
  - IceGrid
  - IceBox
  - IceStorm
  - IcePatch2
  - Glacier2
Freeze

- Object persistence
  - You define what constitutes the persistent state of your objects.
    - This isn't trivial for complex objects containing elaborate data structures.
  - The Freeze compiler generates code that saves/restores that state to a database.
IceGrid

- Provides many useful services...
  - Location service to resolve indirect proxies.
  - Can start servers on demand.
  - Supports replication and load balancing.
  - Automates distribution and patching of servers.
IceBox

- Allows you to package several Ice applications into a single process.
  - Using, for example, DLLs or shared libraries
  - ... or by taking advantage of the features of relevant virtual machines
    - JVM
    - CLR
IceStorm

- A publish/subscribe service
  - Applications can subscribe to “event” categories.
  - When a server publishes the event, every subscriber is alerted.
  - Also called the Observer pattern.
- Decouples clients from servers.
  - Clients don't know the servers... only receive events.
  - Servers don't know the clients.
- Useful when there are a large number of clients.
IcePatch2

- Patch distribution service for clients.
  - Clients connect to an IcePatch2 server.
  - Request updates.
  - Server pushes updated software to the client where it is automatically installed.
Glacier2

• Firewall and security services for Ice
  • Passing Ice traffic through a firewall is problematic.
    – Connections managed by Ice runtime, not application.
    – Ice runtime normally selects ports, etc.
  • Glacier2 allows controllable connection management behavior to facilitate firewall interactions.
  • Also supports encrypted connections, mutual authentication, etc.