

HIVE Kernel: working design 14/1/98

Chris Greenhalgh and Dave Snowdon

Programs, files and classes

Programs are shown at the top level, then files which comprise that program, then classes within those files.

- libhive.so - the core system library, used for all applications
 - BasicTypes.h, .C - generic types
 - Root - abstract base class with destructor and clone() method
 - (Ordinate_t, Radian_t, Status)
 - List.h, .C - list utility classes
 - List : Serializable - a single linked list
 - ListElement : Serializable - an item in a List, include Root* data
 - HashTable.h, .C - hash table utility classes
 - HashTable : Serializable - a hashtable
 - HashTableElement : ListElement - an item in a hashtable (integer hash key)
 - HashTableIterator - iterator for a hash table
 - Util.h, .C - general utility classes
 - Time : Serializable - interface to and representation of time
 - Serializable.h, .C - object serialisation support code
 - ObjectInputStream : Root - abstract base class for object input streams
 - ObjectOutputStream : Root - abstract base class for object output streams
 - Serializable : Root - abstract base class for objects which can be serialized
 - AsciiObjectInputStream : ObjectInputStream - object input from C++ istream, ascii
 - BinaryObjectInputStream : ObjectInputStream - object input from C++ istream, binary
 - AsciiObjectOutputStream : ObjectOutputStream - object output to C++ ostream, ascii
 - BinaryObjectOutputStream : ObjectOutputStream - object output to C++ ostream, binary
 - AsciiFileObjectInputStream : AsciiObjectInputStream - object input from named file, ascii
 - BinaryFileObjectInputStream : AsciiObjectInputStream - object input from named file, binary
 - AsciiFileObjectOutputStream : AsciiObjectOutputStream - object output to named file, ascii
 - BinaryFileObjectOutputStream : AsciiObjectOutputStream - object output to named file, binary
 - Network.h, .C - low-level communications classes (TCP and UDP multicast)
 - Network - process interface to network (multiple sockets)
 - Socket - abstract base class for a network access point (socket)
 - ListenSocket : Socket - TCP server listen socket

- ConnSocket : Socket - TCP connection end-point socket
- GroupSocket : Socket - UDP multicast datagram socket
- Message : ObjectInputStream, ObjectOutputStream - network marshalled message
- NetworkEventHandler.h, .C - callback wrapper for Network
 - NetworkEventHandler - callback wrapper for Network class
- Geometry.h, .C - general geometry-related stuff
 - Point2D : Serializable - 2D point
 - Point3D : Serializable - 3D point
 - Quaternion3D : Serializable - quaternion
 - Matrix3D : Serializable - homogeneous 3D matrix (4x4)
 - Extent3D : Serializable - 3D extent, min and max
 - Trajectory3D : Serializable - time-varying trajectory with start position and orientation, velocity and spin
- Item.h, .C - basic distributed database types
 - ItemID : Serializable - database item identifier
 - ItemData : Serializable - item base class, including id and parent
 - GeometryData : ItemData - geometry item specifying URL
 - EntityData : ItemData - 3D entity item specifying transform and extent
 - AttributeData : ItemData - general text attribute (name, value) pair
 - LinkData : ItemData - link item specifying target id
 - BehaviourData : ItemData - behaviour (code), undefined at present
- Event.h, .C - event stuff, i.e. representations of actions and operations
 - Event : ListElement - a single event, base class with single data item
 - EventQueue : List - a queue of events
 - EventSetOwnerData : Serializable - event data for an item ownership transfer
 - EventSetTrajectoryData : Serializable - event data for a set trajectory event
- Environment.h, .C - environment (replication) stuff
 - EnvironmentID : Serializable - an environment identifier
 - EnvironmentItem : Serializable - an item within an environment, includes ItemData *item
 - EnvironmentItemHashTable : HashTable - the lookup table to find items within an environment
 - Environment : Serializable - a single replicatable environment (database partition)
- AgentID.h, .C - agent identification
 - AgentID : Serializable - agent identifier
- Agent.h, .C - agent, i.e. active process which interacts with environments
 - Agent : Root - an agent or active process
- Trader.h, .C - trader/name-service utilities
 - TraderClient - a trader client with facilities to make requests

- TraderRequest : Serializable - a request to a trader
 - TraderResponse : Serializable - a response from a trader
 - Trader - core trader process functionality
- Trader - the per-machine trader and name-service
 - TraderMain.C
- bodyExample - the example application
 - bodyExample.C
- EchoServer - network-level unicast test program
 - EchoServer.C
- EchoClient - network-level unicast test program
 - EchoClient.C
- GroupTest - network-level multicast test program
 - GroupTest.C
- testSerializable - test of serialization
 - testSerializable.C
- TraderTest - test application for Trader
 - TraderTest.C

There is also a Makefile for convenience of building.

Note that the system currently depends on the CTM and CRG components of Nottingham's CRGSHARE/CVE system. This dependency may be removed in later versions.

Class hierarchy

- Root
 - Serializable
 - List
 - EventQueue
 - ListElement
 - HashTableElement
 - Event
 - HashTable
 - EnvironmentItemHashTable
 - Time
 - Point2D
 - Point3D
 - Quaternion3D
 - Matrix3D
 - Extent3D
 - Trajectory3D

- ItemID
- AgentID
- EnvironmentID
- ItemData
 - GeometryData
 - EntityData
 - AttributeData
 - LinkData
 - BehaviourData
- EventSetOwnerData
- EventSetTrajectoryData
- EnvironmentItem
- Environment
- TraderRequest
- TraderResponse
- ObjectInputStream
 - AsciiObjectInputStream
 - AsciiFileObjectInputStream
 - BinaryObjectInputStream
 - BinaryFileObjectInputStream
 - Message (also ObjectOutputStream)
- ObjectOutputStream
 - AsciiObjectOutputStream
 - AsciiFileObjectOutputStream
 - BinaryObjectOutputStream
 - BinaryFileObjectOutputStream
 - Message (also ObjectInputStream)
- Agent
- HashTableIterator
- Network
- Socket
 - ListenSocket
 - ConnSocket
 - GroupSocket
- NetworkEventHandler
- TraderClient
- Trader

Working notes

Each process or application requires its own instance of the Network and NetworkEventHandler classes to provide the low-level communications interface.

Each process also creates an instance of class Agent to represent itself within the system.

Agent instances can create new Environments or join to existing ones. Each environment is fully replicated and comprises:

- Pending events which have been generated locally or remotely and which have yet to be applied to the environment.
- Sending events which have been generated locally but not yet propagated to replicas of the environment.
- A hierarchy of EnvironmentItems, each of which has zero or one parent items.

The basic item types which make up the environment database are:

- Entity - the root for a 3D entity with a transformation (hierarchically composed) and an extent.
- Geometry - a 3D graphical geometry, currently a URL.
- Attribute - a textual (name, value) pair
- Link - a reference to another item to allow the construction of graphs as well as trees of items
- Behaviour - the specification of a behaviour, currently undefined.

When an Agent joins an Environment it is replicated locally. An Agent can observe and modify an environment through its methods. Modifications to an environment are actually mapped to the creation of appropriate events which are communicated and enacted at a later time.

Known problems and required extensions

- It will only work for 2 users at present because the master process is not fanning out messages from the slaves to other slaves.
- The matrix and point manipulation routines need to be added.
- The environment methods to traverse the database need to be added.
- The support for atomic operations needs coding.
- Support for parent items in different environments need to be added.
- Behaviours need to be defined and supported.
- Trajectories need to be handled (applied to local items).
- Physical rather than logical multicast needs to be employed where appropriate.
- Serialization class name look-up could be generalised (added by hand at present).
- Geometry fetching and handling needs to be added.
- Queueing disciplines (e.g. for causal ordering) need to be defined and implemented.
- Ownership transfer may need work.
- Generic support for multiple environments is required.