Mobile Streams

M. Ranganathan, Laurent Andrey, and Virginie Schall
National Institute of Standards and Technology
Anurag Acharya
University of California, Santa Barbara

Mobile Streams

M.Ranganathan, Laurent Andrey, Virginie Schaal {mranga,andrey,schaal}@nist.gov
National Institute of Standards and Technology, Gaithersburg, MD 20899
Anurag Acharya
Dept. of Computer Science, University of California, Santa-Barbara, CA93106

Abstract

We present a toolkit for building event-driven, re-configurable distributed systems. An event-driven application is driven by asynchronous inputs that cause event-handlers to be invoked. A large class of distributed collaborative, testing, monitoring and control applications fit this paradigm - for example, conferencing and conference control applications, distributed control and testing applications and many others. In each case the notion of an “event” varies. In a collaborative system, events are user inputs; in a distributed monitoring and control system, events are changes in transducer inputs; in a distributed testing scenario, events are test outputs, timer alarms and so on. In such systems, it may be useful to have the ability to dynamically extend and re-configure the system. For example, in a collaborative system, different users may be interested in getting notification of different events that may not be known to the system designer a-priori – necessitating dynamic extension while the system is in execution. There are also situations where system design and performance may be enhanced by dynamic re-configuration - that is dynamic re-mapping of the system functionality while the system is in execution. Our goal is to build a system that enables the scripting of such event-driven applications.

Our basic abstractions consist of Mobile Streams (MStreams) and event handlers. A Mobile Stream is a named communication end-point in a distributed system that can be moved from machine to machine as computation is in progress while maintaining a well-defined ordering guarantee. The closest analogy to an MStream is an “active mobile mailbox”. Like a mailbox, messages may be sent asynchronously to the MStream and are consumed in the same order in which they were sent. By attaching event handlers, message arrivals trigger Handler executions and hence the MStream becomes an “active mailbox”. An MStream has a globally unique name and may be located on any machine that runs an execution environment for it and allows it to be moved there. The ability to move an MStream around makes it a "mobile mailbox". Handlers may be dynamically attached to (and detached from) an MStream and are independently and concurrently invoked for each event. Handlers operate in an atomic fashion. By “atomic” we mean that changes in the state of the MStream (which includes various attributes such as its location, the set of Handlers attached to it and so on) are deferred until the time when the Handlers complete execution. Handlers may append messages to other MStreams - triggering Handler executions at the target MStream when the message is delivered. Handlers are organized into groups (called Agents) with each group having its own interpreter and thread of execution and an MStream may have several Agents associated with it.

A distributed system is organized around its communication end-points i.e. MStreams, and by associating Agents with these end-points, which, in turn, attach Handlers for specific events. An Agent may specify a portion of its global state as being in its *briefcase* – indicating to the system that this state needs to be relocated to the new location when an MStream is moved.

Using the mechanism we have just described, an entire distributed system can be scripted and deployed from a single point of control and dynamically extended and re-configured while it is in execution. To set up such a system, the controller simply defines MStreams, associates handlers with the MStreams and moves the MStreams to the desired locations. This is useful in various scenarios such as web-based testing of distributed systems and conferencing where the web-server can set up the distributed test script for the entire test from a single point of control.

In scenarios involving multi-party collaboration, it is necessary to be able to place controls on how the system may be extended and re-configured. To allow this we have incorporated a resource-control mechanism. Each MStream can be created with its own resource-controller that allows the user to place restrictions on MStream opens, close, movement and so on. The system relies on daemons started at each participating site

to host an execution environment for MStreams. Each of these may be started with a script that permits controls to be placed on the resource usage at that site. In addition, we have a System Resource Controller for the entire system that allows the system designer to place controls on MStream creations, deletions and new peer additions.

Our prototype system called AGNI (Agents at NIST) is a Tcl-based multi-threaded system based on unmodified Tcl 8.1 (a2). Each group of handlers runs its own interpreter and has its own thread. Message delivery to Streams is via a custom reliable peer-to-peer message protocol built on top of UDP that preserves FIFO delivery order despite dynamic re-configuration. AGNI is currently operational and has been used to implement three fairly substantial applications – a toolkit for collaboratively sharing unmodified Tk applications (adapted from the TK-Replay code developed by Charles Crowley [Cr95]), a debugger for MStream programs (based on the freely available Tcl-Debug debugger developed by Don Libes [Li93]) and a monitor for visualizing an MStreams-based distributed system. Using our Tk-sharing toolkit, we were able to share a Tk-Drawing application[T98]. We have also developed a closely related application to record the interactions that occur in a collaborative conference and replay these interactions later. We have applied this technology to record and replay the user-GUI interactions that occur during the use of the MITRE XCVW collaborative tool. We are currently in the process of building a web-based tester for peer-to-peer applications that uses Mobile Streams as a run-time system. Other applications we are considering include a collaborative system for viewing of images from a large image data repository in telepathology applications and support for mobile proxys in ubiquitous computing applications.

References

[Cr95] C. Crowley, "Tk-Replay: Record and Replay in Tk", *USENIX Third Annual Tcl/Tk Workshop*, 1995.

[L93] D. Libes, "A debugger for Tcl " *Tcl/Tk workshop, 1993.*

[T98]http://www.inftechnik.tu-ilmenau.de/~silvio/research/soft.htm
Tk-Draw web page.