The Active Streams Approach to Adaptive Distributed Systems

Fabián E. Bustamante, Greg Eisenhauer, Karsten Schwan, and Patrick Widener
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332, USA
{fabianb, eisen, schwan, pmw}@cc.gatech.edu

The explosive growth of the Internet, with the emergence of new networking technologies and the increasing number of network-capable end devices, is paving the way for a number of novel distributed applications and services. Cooperative distributed systems have become a common computing model, and pervasive computing has caught the interest of academia and industry. In the high-performance community, for example, distributed scientific collaboration projects [9, 8] aim to create environments where geographically dispersed communities of scientists cooperate to perform experiments on remote instruments and to share and discuss their findings. The growing importance of pervasive computing [12], with its goal of providing access to information anywhere/anytime through an invisible computing infrastructure, is reflected in the number of major academic research endeavors [10, 4, 3] and the increasing interest from industry [6, 7].

The realization of these types of applications is complicated by the characteristics of their target environments, including their heterogeneous nature as well as the dynamically varying demands on and availability of their resources. Dynamic variations in resource usage are due to applications' data dependencies and/or users' dynamic behaviors, while the run-time variation in resource availability is a consequence of failures, resource additions or removals, and most importantly, contention for shared resources.

To support future network applications, we believe that new services need to be customizable, applications need to be dynamically extensible, and both applications and services should be able to adapt to variations in resource availability and demand. A comprehensive approach to building new distributed applications can facilitate this by considering the contents of the information flowing across the application and its services and by adopting a component-based model to application/service programming. It should provide for dynamic adaptation at multiple levels and points in the underlying platform; and, since the mapping of components to resources in dynamic environment is too complicated, it should relieve programmers of this task. In this paper we propose *Active Streams* [2], a middleware approach and its associated framework for building distributed applications and services that exhibit these characteristics.

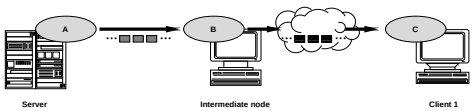

Figure 1. Making a stream active.

With Active Streams, distributed systems are modeled as being composed of *applications*, *services*, and *data streams*. Services define collections of operations that servers can perform on behalf of their clients. Data streams are sequences of self-describing application data units flowing between applications' components and services. They are made *active* by attaching application- or service-specific location-independent functional units, called *streamlets* (Figure 1).

Streamlets are self-contained units that operate on records arriving on their incoming streams and generate records placed onto their outgoing streams. Streamlets can be obtained from a number of locations; they can be downloaded from clients or retrieved from a streamlet repository. Streamlets are created using E-Code, a subset of a general procedural language, and dynamic code generation is used to insure that they can be dynamically deployed and efficiently executed across heterogeneous environments. E-Code's dynamic code generation capabilities are based on Icode/Vcode [11]. Icode/Vcode supports dynamic code generation for MIPS, Alpha and Sparc processors, and has been extended to support MIPS n32 and 64-bit ABIs, Sparc 64-bit ABI, and x86 processors. Legacy applications, as well as those built with new approaches such as CCA [1], can be easily integrated with Active Streams because of our focus on stream-based transformations on the datapath

Application evolution and/or a relatively coarse form of adaptation are obtained by the attachment/detachment of streamlets that operate on and change data streams' prop-

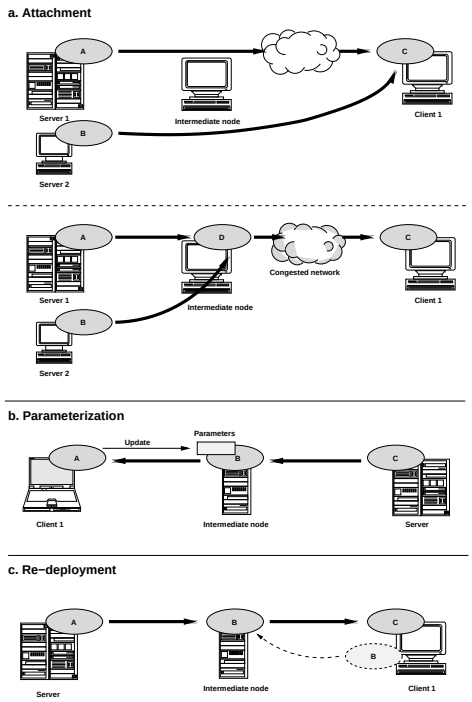

Figure 2. Types of adaptation in Active Streams.

erties. Finer grain adaptation involves tuning an individual streamlet's behavior through parameters remotely updated via a push-type operation, and by re-deploying streamlets to best leverage the dynamically-changing available resources over the datapath (Figure 2).

Active Streams are realized by mapping streamlets and streams onto the resources of the underlying distributed platform, seen as a collection of loosely coupled, interconnected computational units. These units make themselves available by running as Active Streams Nodes (ASNs), where each ASN provides a well-defined environment for streamlet execution. Active Streams applications rely on a push-based customizable resource monitoring service (ARMS) to collect resource information and trigger adaptation. Through ARMS, applications can select a subset of the data made available by distributed monitors. These data streams can be integrated to produce application-specific views of system state and decide on possible adaptations.

As is common in distributed systems, a directory service provides the "glue" that holds the Active Streams framework together. The dynamic nature of most relevant objects in Active Streams makes the passive client interfaces of classical directory services inappropriate. Thus, the Active Streams framework includes a *proactive* directory service with a publish/subscribe interface through which clients can register for notification on changes to objects currently of interest to them. The levels of detail and granularity of these

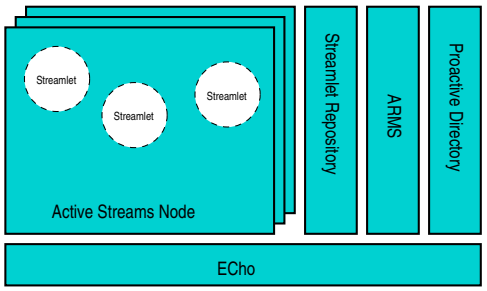

Figure 3. Active Streams Framework.

notifications can be dynamically tuned by the clients.

These three components, together with the repository service previously mentioned, conform the core of the Active Stream framework (Figure 3). The whole framework relies on ECho, a high performance event infrastructure [5], for its communication requirements.

The implementation of Active Streams is mostly complete, and we plan on making it available by December 2001.

References

[1] R. Bramley, K. Chiu, S. Diwan, D. Gannon, M. Govindaraju, N. Mukji, B. Temko, and M. Yechuri. A component based services architecture for building distributed applications. In *Proceedings of the 9th High Performance Distributed Computing (HPDC-9)*, Pittsburgh, PA, August 2000.

[2] F. E. Bustamante and K. Schwan. Active Streams: An approach to adaptive distributed systems. Tech. report, College of Computing, Georgia Institute of Technology, Atlanta, GA, June 1999.

[3] CMU. Project Aura. http://www.cs.cmu.edu/ aura.

[4] M. Dertouzos. The oxygen project. *Scientific American*, 282(3):52–63, August 1999.

[5] G. Eisenhauer, F. E. Bustamante, and K. Schwan. Event services for high performance computing. In *Proceedings of the 9th High Performance Distributed Computing (HPDC-9)*, Pittsburgh, PA, August 2000.

[6] IBM. Pervasive computing. http://www-3.ibm.com/pvc.

[7] D. Neel. Compaq's CEO advocates pervasive computing applications. *InfoWorld*, March 26 2001.

[8] NPACI. Alpha projects. http://www.npaci.edu/Alpha, 2001.

[9] NSF:NEES Project. NEESgrid: earthquake engineering virtual collabortory. http://www.neesgrid.org, 2001.

[10] G. I. of Technology and the Oregon Graduate Institute. Infosphere: Smart delivery of fresh information. http://www.cc.gatech.edu/projects/infosphere.

[11] M. Poletto, D. Engler, and M. F. Kaashoek. tcc: A template-based compiler for 'c. In *Proceedings of the First Workshop on Compiler Support for Systems Software (WCSSS)*, February 1996.

[12] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.