

Position Summary: The Importance of Good Plumbing

Reconsidering Infrastructure in Distributed Systems

Andrew Warfield and Norm Hutchinson
University of British Columbia
{andy,norm}@cs.ubc.ca

The fundamental abstractions and mechanisms used to deliver communications within distributed systems have remained essentially unchanged for a considerable time. With very few exceptions, operating systems implement a simple, socket-based approach to communications in which the host's involvement in a particular communication stream ends at the network interface. TCP/IP provides an environment in which the notion of a data stream does not actually exist within the network, but rather is an abstraction made by 'connected' endpoints.

Above the network, significant developments have been made to advance the state of distributed systems technology. Many sizeable middleware infrastructures have been developed and are actively being used in the construction of commercial distributed applications. Despite the benefits provided by these packages, they remain dependent on an insufficient infrastructure, which may be considered according to three fundamental flaws: 1. TCP/IP simply does not provide adequate network functionality for distributed systems. The shortcomings of the protocol provide a list of ongoing research problems including mobility, quality of service, and group collaboration. 2. The primary OS abstraction for a stream, the socket, is inflexible and represents a poor coupling between the network and the OS. 3. Remote procedure calls, which are the *de facto* approach to distributed invocation almost universally attempts to hide the network, obscuring failures (and features) from overlying applications.

Within the network TCP/IP also proves problematic. Considerable research efforts exist in the ongoing attempts to carry a legacy protocol well beyond the scope of its initial design. Traffic management, congestion control, and resource reservation all remain largely unsolved problems due to the difficulties of managing data streams within the network.

The existing infrastructure successfully provides a serviceable network. We feel that the ongoing functionality of the existing system explains the thrust of research towards addressing individual deficiencies rather than addressing the system as a whole. However, there is an opportunity to realize substantial benefits through the development on an abstraction that is understood and supported by both the OS and the network routers. Our work to date has been in the design and development of a stream-centric model for communications which we have called the *flow* [1]. A flow is a uniquely named, message-based, multicast communications stream.

Flows are named by FlowIDs which represent a collection of resources used to provide a communications stream in the same manner that process IDs represent resources associated with a computational task. By providing distinct names for these multicast streams, services become decoupled from network endpoints. This single property provides sweeping benefits for mobility, fault tolerance, and resource location.

In addition to naming, we have implemented three properties of flows which we feel are beneficial to distributed applications. First, flow messages are *banded*. Each flow has a label space of 128 bands within which messages may be sent. This allows an external separation of concerns for messages within a stream, and also provides an effective means to tie administrative and fault messages to a stream from points within the network. Second, flows support a notion of *locality*. Locality acts as an extension of TTL that allows message transmission to be scoped according to criteria such as geographic area, available bandwidth, or latency. Finally, flow messages are delivered from the network to client-defined queues. These queues allow local delivery options, such as drop strategy and message ordering, to be defined and implemented at the application. Queues may be attached to various bands of a flow, providing a great degree of flexibility in how and where the message stream is used.

We have finished an initial implementation of flows as a network middleware and have become convinced that they are an interesting and useful communications abstraction. We are currently extending our definition to allow the recursive embedding of flows, providing a hierarchy of streams. We feel that this property will be very beneficial both in terms of traffic management and software design. Additionally, we are investigating methods of typing individual data streams in order that the format of their content may be advertised to devices along the transmission path. Our investigation of these two properties coincides with efforts to implement flows efficiently at the network layer.

References

- [1] Flows project web page. www.cs.ubc.ca/spider/andy/flows/.

We would like to acknowledge Alexander Fraser and Glenford Mapp at AT&T research for their support and ideas, in particular that of recursive flows.