ATP: AUTONOMOUS TRANSPORT PROTOCOL

Tamer Elsayed[1]*, Mohamed Hussein*[1]*, Moustafa Youssef*[1]*, Tamer Nadeem*[1]*,*
Adel Youssef[1]*, and Liviu Iftode*[2]

Abstract - *In this paper we present the design of the Autonomous Transport Protocol (ATP). The basic service provided by the ATP is to maintain a reliable transport connection between two endpoints, identified by content identifiers, independent of their physical locations. Autonomy allows dynamic endpoints relocation on different hosts without disrupting the transport connection between them. The ATP depends on the existence of an underlying enhanced content-based network to achieve its goals. Data is transferred by a combination of active and passive operations, where the ATP layer of a node can decide whether to actively push the data to the destination or to passively wait for the destination endpoint to pull the data. The decision to use either the active mode or the passive mode can be taken by a local policy on the node running the ATP.*

Index Terms - Autonomous transport protocol, Content-based networks, Instance-based networks, Mobility management, Peer-to-peer systems, Transport protocols, Ubiquitous computing

INTRODUCTION

As ubiquitous computing emerges; the users, not the end hosts, should become the focus of the communication. To achieve this goal, connections should be carried between the users, independent of the hosts on which they are located. Content-based networks (CBN) provide a mechanism to map a content to a specific host of a peer-to-peer (P2P) network, and to query the location of this content. Considering a user's endpoint as a content, a transport protocol over a CBN uses the lookup service mechanism to locate the users' endpoints. The challenge is how to maintain a reliable connection between the users' endpoints as they roam in the environment moving from one host to another.

In this paper, we introduce the Autonomous Transport Protocol (ATP) that allows dynamic endpoint relocation without disrupting the connection between them. The ATP has the following features:

- It does not enforce any naming scheme on the user application. The application is responsible for uniquely identifying the endpoint.
- The endpoints of a transport connection are defined as contents in the content-based network. This decouples the connection from the physical host where the user endpoint is located, and hence ensures autonomy.
- Mobility of the endpoints is handled by dynamically changing the mapping between the endpoints and the hosts using the new instance-based network (IBN) in which we enhanced the content-based network to provide additional functionalities, as we will describe later. The ATP layer is responsible for moving data segments to the destination and the acknowledgments back to the source regardless of their current location in the network.
- Data is transferred by a combination of active and passive operations, where the ATP layer of a node can decide whether to actively push the data to the destination or to passively wait for the destination endpoint to pull the data. The decision to use either the active mode or the passive mode can be taken by a local policy on the node running the ATP.

Figure 1 shows the system architecture for an ATP environment. The ATP protocol stack consists of four layers: the underlying-network layer, the IBN layer, the ATP layer, and the application layer.

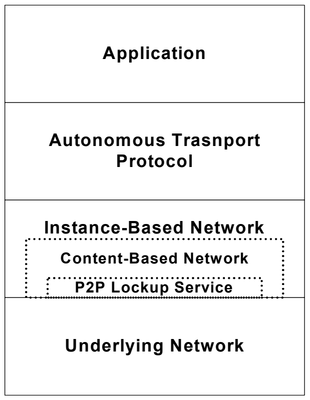

FIGURE 1
SYSTEM ARCHITECTURE

Example Scenario

Figure 2 shows an application scenario for the ATP. A set of wireless nodes, represented by small black circles, is distributed over a certain area. An observer *Src* generates statistical data about the environment while it is moving around. This data is transmitted reliably to an analyzer *Dst*.

The observer triggers the tracking application on the nearest node, *Node*$_1$ in Figure 2, to become active and to

[1] Department of Computer Science, University of Maryland, College Park, MD 20742, USA, {telsayed, mhussein, moustafa. nadeem, adel}@cs.umd.edu
[2] Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA, iftode@cs.rutgers.edu

transmit the collected data, through an ATP connection, to the analyzer at *Node$_3$*, as in the figure. During the observer movement, it can become out of the vicinity of any tracking node. For example, as in Figure 2, while the observer moves from *Node$_1$* to *Node$_2$* (source migration), it is, temporally, not in the vicinity of any tracking node. Once the observer becomes out the vicinity of *Node$_1$*, the tracking application, on *Node$_1$*, stops tracking, however, the ATP layer on *Node$_1$* continues handling the established connection by reliably transmitting the remaining data to the analyzer. When the observer becomes within the vicinity of *Node$_2$*, it triggers the tracking application on that node to transmit the collected data to the analyzer using the already established ATP connection. Meanwhile, the ATP layer on *Node$_1$* may still be transmitting the rest of the data it has to the observer.

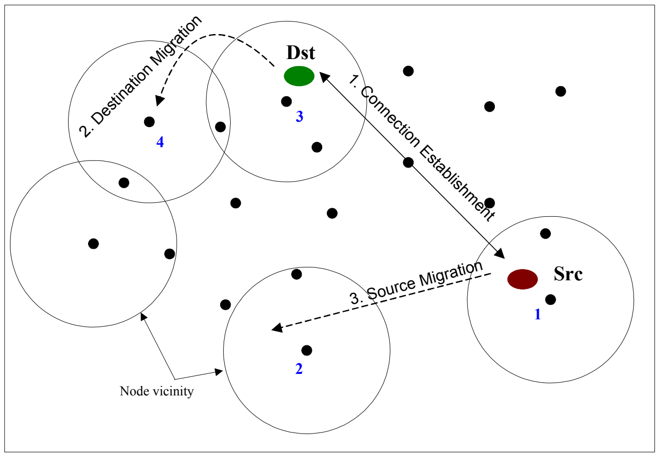

FIGURE 2
A TYPICAL ATP APPLICATION SCENARIO

Similarly, the analyzer changes the monitoring node from *Node$_3$* to *Node$_4$* (destination migration) as in Figure 2. While the analyzer is moving from *Node$_3$* to *Node$_4$*, the ATP layer at *Node$_3$* continues handling the connection on behalf of it. When the analyzer is setup on *Node$_4$*, the connection is updated to forward the data to the new physical location of the analyzer. The ATP layers at *Node$_3$* and *Node$_4$* are responsible for transferring the data segments received at *Node$_3$*, during the migration of the analyzer, to *Node4*. Due to the presumed spatial locality of the observer's movement, transferring the data from *Node$_3$* to *Node$_4$* has lower overhead than retransmitting the data from the observer. In this scenario, the observer and the analyzer should not be aware of the physical locations of each other and the movement handling should be transparent to both of them.

Roadmap

This paper is organized as follows: in the following two sections the two main layers in our architecture (the IBN layer and the ATP layer) are explained. Next, our implementation status is presented. Then, related work is surveyed. Finally, our conclusion is provided.

INSTANCE-BASED NETWORK (IBN) LAYER

We define content-based network (CBN) as a network of endpoint entities called *contents* where each content is addressed or located by its name, properties or attributes, independent of its physical location. The content could be a user, an application service, a document, a network node, a network connection or any other object. Unlike IP networks where the IP address is not just an identifier but also a locator, CBN addressing is decoupled from the location of contents. Contents can actively communicate with each other by sending or receiving messages, or performing a lookup for other contents. Other content types, such as a document, can be passively stored in the network.

The CBN layer extends the functionality provided by the current peer-to-peer lookup services (such as CAN[1], Chord [2], Pastry [3], and Tapestry [4]). Peer-to-peer lookup services provide a mechanism to map a key to some node, specified by the lookup service, in the network and allows the user to query for these keys. The CBN, however, maps a content to a *specific* node in the network and routes messages to this node.

We have developed an enhanced content-based network named as the *instance-based network (IBN)*. The IBN has the unique feature of allowing different instances of the same content to be stored in the network (and hence the name IBN).

A content of the IBN is addressed using a name *X* and an instance identifier ($i_1,i_2,\ldots,i_n$), where $i_1, i_2, \ldots, i_n$ are *n* integer numbers. We use the notation $X: i_1,i_2,\ldots,i_n$ to refer to an instance of a content *X*. The semantics and dimensionality (*n*) of the instance identifier is assigned by the user of the IBN. These semantics include the ordering relation between different instances. For example, in a file archiving system, a file name can be represented as logfile:1,0,1 to represent the version 1.01 of the file logfile. These semantics are assigned by the file archiving system.

Routing in the proposed IBN is instance-based. A message destined to content $X: i_1,i_2,\ldots,i_n$ is routed to the content with the same content name *X* and with the highest published instance identifier that is less than or equal to ($i_1,i_2,\ldots,i_n$). For example, if we have *X:3*, *X:5*, and *X:8* as all the published content instances with content name *X*; and a message is sent with destination content *X:7*, this message is routed to the content *X:5*.

AUTONOMOUS TRANSPORT PROTOCOL

A connection in the ATP is established between two endpoints that are identified by their content ID's. Endpoints could migrate or temporarily disappear from the network while data segments and acknowledgments continue to flow between them. We assume that all connections are simplex. Extension to the full-duplex case is straightforward.

Assume a source endpoint *Src*, attached to node *Node$_1$*, establishes an ATP connection with a destination endpoint

Dst that is attached to *$Node_3$* (Figure2). When *Src* migrates to a new node *$Node_2$*, the ATP layer in *$Node_1$* spawns an agent that takes care of sending any data in the sending buffer of *Src* and receiving acknowledgments. The ATP layers on *$Node_1$* and *$Node_2$* cooperate to make the migration transparent to *Dst*. Similarly, when Dst migrates to a new node *$Node_4$*, the ATP layer on *$Node_3$* spawns an agent that acts on behalf of *Dst* to buffer any received data and to send acknowledgments. The ATP layers on *$Node_3$* and *$Node_4$* cooperate to make the migration transparent to *Src*. The migration step can be performed multiple times and there can be multiple agents working for the same endpoint at any time.

An ATP agent can take the decision to participate actively in the connection by pushing data segments to the destination or to wait passively for the destination to pull them. In the former case, the node publishes (registers) itself in the network as *AS* (Active Source), while in the latter case, the node publishes itself as *PS* (Passive Source). The default mode of the agents acting on behalf of the source is the active mode. Agents acting on behalf of the destination should buffer the received data until the destination appears on a new node. Therefore, these agents stay in the passive mode until the destination reappears and pulls the buffered data.

Each agent has a unique name composed of the original content ID plus the starting sequence number of the data it is responsible for. For example, *AS:4* denotes an agent for the source endpoint in the active mode responsible for data segments starting from sequence number 4.

In the following subsection, details of connection establishment, operation of the ATP in the basic mode (when no migration takes place), and a simple source migration scenario are explained. Details of destination migration and more complicated scenarios can be found in [5].

Connection Establishment

The connection establishment uses 3-way handshaking similar to the TCP handshaking. Initially both ends, *Src* and *Dst* publish themselves in the network by publishing their IDs as in Figure 3. The content ID is used as the address in the communication. Once the connection is established, both ends will have a spawned ATP agent (*Ag1* and *Ag3*) and the communication goes through them.

Basic Mode

In this mode, neither *Src* nor *Dst* migrates. Operation in this mode is similar to the operation of the TCP over the IP networks. Data segments (*$Data_1$*; *$Data_2$*; …) generated by *Src* are sent by *Ag1* with destination address *D:0* using the *Send* function as in Figure 3. A cumulative acknowledgment from *Ag3* for sequence number *k* (*Ack_k*), where $k \geq 0$, is sent to *AS:k* which is routed to *$Node_1$* by the underlying IBN.

Source Migration

Figure 3 shows an example of the ATP operation when the *Src* migrates from *$Node_1$* to *$Node_2$*.

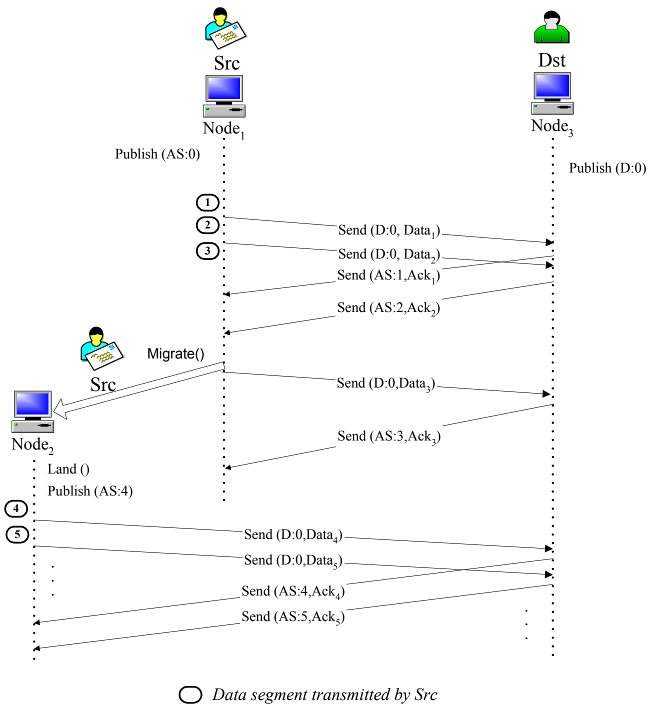

FIGURE 3
A SOURCE MIGRATION SCENARIO

Upon migration: *Src* informs *Ag1* of its intention to migrate by invoking the ATP function *Migrate*. Upon the invocation of this function, *Ag1* stores the current status of the connection in the IBN. The status contains the last sequence number buffered in the ATP layer on *$Node_1$*.

During migration: The content *AS:0* is still published in the IBN and attached to *Ag1* at *$Node_1$*. *Ag1* is responsible for transmitting the remaining data in the sending buffer, accepting acknowledgments from the destination, managing the transmission window and retransmitting segments if necessary.

Upon landing: When *Src* lands on *$Node_2$*, it invokes the ATP function *Land* which spawns a new ATP agent *Ag2* that is responsible for the connection on the new node. *Ag2* starts by restoring the status of the connection from the IBN. *Ag2* publishes a new content in the IBN with ID *AS:j*, where *j* is the sequence number obtained from the stored status ($j = 4$ in Figure 3), and attaches this content to *$Node_2$*.

After landing: *Src* starts sending data. A cumulative acknowledgment from *Ag3* for segment *k* is sent with destination address *AS:k*. If *k* is less than *j*, the acknowledgment is routed to *Ag1* (using the instance-based routing feature of the underlying IBN). Similarly, if *k* is greater than or equal to *j*, the acknowledgment is routed to *Ag2*. The multiple migrations case is a direct extension to the single migration case [5].

Implementation

We have implemented a prototype of the ATP protocol over Pastry [3] as a P2P overlay network layer. The prototype is deployed over a set of independent nodes at University of Maryland. A simple ATP-aware application was implemented and was run on each node of the network.

Several design issues are still under investigation. Those issues include mechanisms for: node switching between active and passive modes, reclaiming network resources, and fault tolerance.

Related Work

Although the mobile IP protocol [6] provides a solution to the host mobility between different networks, a user is bound to a single host during the lifetime of a connection. In [8], [9] a mobility solution at the TCP level, that allows the connection to remain open as the host moves between different networks, is provided. However, both solutions do not allow the endpoint to change hosts.

Similar work to the proposed IBN is the Internet Indirection Infrastructure (*I3*) [6] which builds an indirection layer, over the Chord peer-to-peer system, that allows a key to be mapped to a specific node in the network. Our IBN is unique in allowing different instances of the same content and in using instance-based routing.

Authors in [10] propose a mobility infrastructure, based on *I3* that offers a rendezvous-based communication abstraction. Although their system is similar to ours, there are significant differences between them. Since their system relies on TCP, it inherits its problems regarding mobility. The *I3* system is explicitly based on the IP routing between the rendezvous point and the target while our proposed protocol does not assume a specific underlying infrastructure. The Mobile Tapestry system [11] offers a similar system to the *I3*-based system. The main difference between a mobile system built on *I3* and Mobile Tapestry is that, while the former provides a single level of indirection as a rendezvous point for redirecting packets, Mobile Tapestry provides multiple points of indirection. This system has the same shortcomings as the I3-based system.

The system in [12] provides a seamless service platform in which endpoints can migrate between different nodes. The system is based on seamless-proxies that form a network between themselves and work as a middleware between the application and the transport layer. Our system differs from this system in that their system is IP-based. Also, for each possible application their system requires an application specific module to handle endpoint migration.

Conclusion

In this paper, we presented the design of the Autonomous Transport Protocol (ATP) that provides a reliable communication between two endpoints regardless of their physical location in the network. We presented an instance-based network (IBN) that provides the ability of having different instances of the same content and the flexibility to map a content to a particular node.

The ATP allows the endpoints to migrate between different hosts transparently to each other. End points spawn ATP instances (agents) as they migrate between network nodes. These agents become responsible for the connection at the corresponding nodes. ATP uses the underlying IBN to route data segments and acknowledgments to the correct agent. Data is transferred by a combination of active and passive operations, where the ATP layer of a node can decide whether to actively push the data to the destination or to passively wait for the destination endpoint to pull the data.

The ATP is essential to applications requiring reliable communication with endpoints migration. We believe that the ATP is important for ubiquitous computing to become a reality.

References

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," in Proceedings of ACM SIGCOMM 2001, 2001.

[2] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications," in Proceedings of ACM SIGCOMM 2001, San Diego, September 2001.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," in Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, November 2001, pp. 329–350.

[4] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," Tech. Rep. UCB/CSD-01-1141, UC Berkeley, April 2001.

[5] T. Elsayed, M. Hussein, M. Youssef, T. Nadeem, A. Youssef, and L. Iftode, "ATP: Autonomous Transport Protocol," Tech. Rep. UMIACSTR-2003-52 and CS-TR-4483, University of Maryland, May 2003.

[6] C. Perkins, "IP Mobility Support," RFC 2002, October 1996.

[7] I. Stoica, D. Adkins, S. Zhaung, S. Shenker, and S. Surana, "Internet Indirection Infrastructure," in Proceedings of ACM SIGCOMM 2002, Pittsburgh, PA, August 2002, pp. 73–86.

[8] B. Zhang, B. Zhang, and I. Wu, "ETCP: Extended TCP for Mobile IP Support," Internet Draft, 1998.

[9] A. C. Snoeren and H. Balakrishnan, "TCP Connection Migration," Internet Draft, November 2000.

[10] S. Zhuang, K. Lai, I. Stoica, R. Katz, and S. Shenker, "Host Mobility Using an Internet Indirection Infrastructure," in Proceedings of MobiSys 2003, 2003.

[11] B. Y. Zhao, A. D. Joseph, and J. D. Kubiatowicz, "Supporting Rapid Mobility via Locality in an Overlay Network," Tech. Rep. UCB/CSD-02-1216, UC Berkeley, November 2002.

[12] K. Takasugi, M. Nakamura, S. Tanaka, and M. Kubota, "Seamless Service Platform for Following a User s Movement in a Dynamic Network Environment," in Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 03), March 2003.