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 whether to use the active or passive modes 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 users, independent from the host 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 layer protocol over a CBN uses the lookup service mechanism to locate the user’s endpoints. The challenge is how to maintain a reliable connection between the user’s endpoints as they roam in the environment moving from one host to another.

In this paper, we introduce the Autonomous Transport Protocol (ATP) which allows dynamic endpoint relocation without disrupting the transport 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 whether to use the active or passive modes 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.
implementation status is presented. Then, related work is surveyed. Finally, 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 [2], Chord [3], Pastry [4], and Tapestry [5]). 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 developed an enhanced content-based network named as the instant-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 \( (i_1, i_2, ... i_n) \), where \( i_1, i_2, ... , i_n \) are \( n \) integer numbers. We use the notation \( X: i_1, i_2, ... , i_n \) to refer to an instance of a content \( X \). The semantics and dimensionality \( (n) \) of the instant identifier tuple 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 \( X: i_1, i_2, ... , i_n \) 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 \( i_1, i_2, ... , i_n \). 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 which are identified by their content ID's. Endpoints could migrate or temporarily disappear from the network while data segments and acknowledgments should continue to flow between them. We assume that all connections are simplex. Extension to the full-duplex 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 \( Node_3 \) (Figure 2). 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 \( Node_3 \) 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 the data or to wait passively for the data to be pulled by the destination. In the former case, the node publishes (registers) itself in the network as \( AS \) (Active Source), while in the latter 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 the segments starting with sequence number 4.

**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 2. The content ID is used as the address in the communication. Once the connection is established, both ends will have a spawned ATP agent (\( Ag_1 \) and \( Ag_3 \)) 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 \( Ag_1 \) with destination address \( Dst \) using the \( Send \) function as in Figure 2. A cumulative acknowledgment from \( Ag_3 \) 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 2 shows an example of the ATP operation when the \( Src \) migrates from \( Node_1 \) to \( Node_2 \).

---

Protocols, Algorithms & Mechanisms
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 Node1.

During migration: The content $AS:0$ is still published in the IBN and attached to Ag1 at Node1. 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 Node2, 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 identifying the agent responsible for maintaining the connection on Node3. The content published by Ag2 has an ID of $PS:j$, where $j$ is the first sequence number not consumed by the application ($j = 1$ in Figure 2), and attaches it to Node3. This content identifier will be used by the ATP agent on Node4 to pull the segments stored by Node3 during the migration. Note also that this agent will be in the passive mode since there is no benefit of actively sending segments to the destination when it is in migration.

Destination Migration

Figure 3 shows the operation of the ATP when Dst migrates from Node3 to Node4.

Upon migration: Dst informs Ag3 of its intention to migrate by calling the function Migrate. In response: Ag3 stores the current status of the connection in the IBN. The status includes the last sequence number consumed by Dst on Node3 and the last acknowledged sequence number. Then, Ag3 publishes a new content in the IBN identifying the agent responsible for maintaining the connection on Node3. The content published by Ag3 has an ID of $PS:j$, where $j$ is the first sequence number not consumed by the application ($j = 1$ in Figure 3) attaching it to Node3. This content identifier will be used by the ATP agent on Node4 to pull the segments stored by Node3 during the migration. Note also that this agent will be in the passive mode since there is no benefit of actively sending segments to the destination when it is in migration.

During migration: Since the content $D$ is still published in the IBN and is attached to Node3, data segments sent from Ag1 is routed to Ag3 on Node3. Ag3 is responsible for buffering the new segments and acknowledging them.

Upon landing: When Dst lands on Node4, it informs the ATP layer of its arrival using the ATP function call Land. In response, ATP spawns an agent (Ag4) to be responsible for this connection. Ag4 starts its operation by performing two
Protocols, Algorithms & Mechanisms

tasks: Restoring the state of the connection and republishing the content D and attaching it to Node3.

After landing: From now on, data segments sent by A/I are routed to Node3. When Dst asks for data, Ag4 starts pulling data segments. A pull request for segment with sequence number k (Pull(k) as in Figure 3) is sent to destination address PS:k that is routed by the IBN to the appropriate node. For example, in Figure 3, a pull for segment 1 is routed to Node3. In response to a pull request, Ag3 replies by sending the data segment with the pulled sequence number. Ag4, also, receives normal segments from AgI, which is still in the active mode. For more details about the single and the multiple migrations case, refer to [6].

IMPLEMENTATION

We have implemented a prototype of the ATP protocol over Pastry [4] 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 has been implemented to be run on each node of the network.

Several design issues are still under investigation. Those issues include: the node switching mechanism between active and passive policies, mechanisms for reclaiming network resources, efficient acknowledgement mechanisms, fault tolerance, security, and end-to-end semantics.

RELATED WORK

Although the mobile IP protocol [1] 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) [7] 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 bases explicitly 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 and has the same shortcomings of it.

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