Instance-Based Networking: A Communication Paradigm for Mobile Applications

Tamer Elsayed, Mohamed Hussein, Moustafa Youssef, Tamer Nadeem, Adel Youssef, Liviu Iftode
Department of Computer Science and UMIACS
University of Maryland
College Park, Maryland 20742
{telsayed,mhussein,moustafa,nadeem,adel,iftode}@cs.umd.edu

Abstract— **In this work we consider the design principles of the *Instance-Based Network* (IBN), an extended version of a generic Content-Based Network(CBN). IBN acts as an overlay communication platform over which end-point entities, called *contents*, communicate independently from their physical locations while providing the flexibility of having different instances of the same content. The semantics of different instances are assigned by the application using the IBN. Routing in the IBN is instance-based; the IBN can route a message to a specific content instance or to the nearest instance, if no exact match is found for the destination content instance. Moreover, the IBN replicates the stored contents in order to provide fault tolerance and IBN nodes along the query path can cache a content to provide fast answers to future queries.**

I. Introduction

Consider a file archiving system over a peer-to-peer network. Files in this system are defined by content identifiers and the system keeps track of different versions of the same file. A user of such a system can request to retrieve a specific version of the file or can request the latest version stored in the system. The file archiving system is an example of a larger class of peer-to-peer applications where entities (files in the file archiving system) are defined by content identifiers (file name) and different instances (versions in the file archiving system) from the same content can exist at the same time. Other examples of applications in the same class include:

- peer-to-peer anycasting where a service is defined by a content ID (service name) and different instances of the same service represent nodes offering the same service. The instance identifier is used to select the closest node to the requesting node depending on some metric.
- A pervasive environment where application endpoints are defined by contents IDs. Applications can migrate from one node to the other and the established communication connections should continue transparently without interruption. Different agents from the same application (instances) works on behalf of the application on different nodes to maintain the connection [1].

We propose the *Instance-Based Network (IBN)* as a communication paradigm to support this class of applications. In an IBN, endpoint entities called *contents* are 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 a unique ID but also a locator, IBN 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.

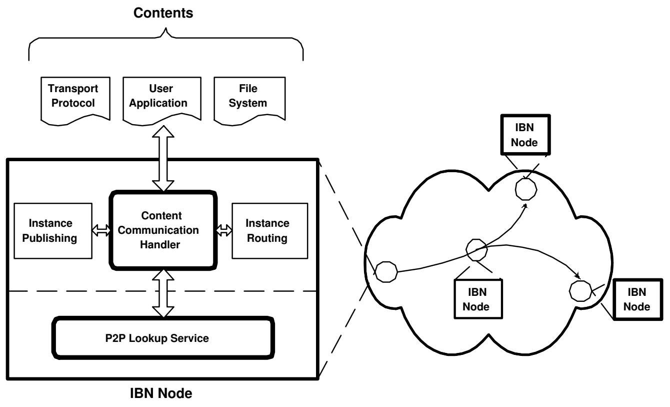

Fig. 1. IBN node architecture

The IBN approach extends the functionality provided by the current peer-to-peer lookup services (such as CAN [2], Chord [3], Pastry [4], and Tapestry [5]) by allowing different instances of the same content to be published in the network (and hence the name IBN). It supports the following functionalities:

- *Instance-node mapping*: The IBN user can ask the IBN to map an instance to a particular node (through a publish operation). All mapping are leased, that is, if the user does not refresh the lease before it expires, the content is removed (or unpublished) from the IBN. Moreover, the user can ask the IBN to freely publish an instance according to a policy of the IBN to achieve a specific objective (like load balancing).
- *Instance communication*: Active endpoints can send messages to other instance-identified endpoints.
- *Instance-based routing*: The IBN can route a message

to a specific content instance or to the nearest instance (the neighborhood metric is discussed below) if no exact match is found for the destination content instance.

- *Replication*: The IBN replicates the stored contents in order to provide fault tolerance.
- *Caching*: Nodes along the query path can cache a content to provide fast answers to future queries.

Figure 1 illustrates the architecture of an IBN node model showing its main components.

II. Addressing

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

III. Routing

The routing in the proposed IBN network is instance-based. A message destined to content $(X : i_1, ..., i_n)$ is routed to the closest published instance of the same content X to the destination instance. The *Closest* semantics are assigned by the application using the IBN and represent a possible ordering relation between different instances. For example, in a file archiving system, they are used for comparing file versions.

IV. Implementation

We propose an implementation that extends the current peer-to-peer lookup services. When the user asks the IBN to publish the content instance $(X : i_1, ..., i_n)$ on a certain node A, the IBN uses the underlying peer-to-peer lookup service to find the node B where the content X should be mapped to. The IBN layer uses the node B to store a mapping between the instance $(X : i_1, ..., i_n)$ and the node A in a data structure $\mathbb{D}$. Note that node B will be responsible for all instances of the same content X.

If an application wants to send a message to an instance $(X : j_1, ..., j_n)$, the IBN routes the message using the underlying peer-to-peer network to the node B. The IBN node B checks the data structure $\mathbb{D}$ to find the nearest instance to the destination instance identifier $(j_1, ..., j_n)$ and the node C where this instance is published (using the routing algorithm is section III). The IBN finally forwards the message to node C. Other IBN services, described in [1], can be implemented in a similar way.

The proposed implementation has the following features:

- A single node is responsible for all instances of a particular content. This allows for efficient algorithms for finding the nearest instance to a given instance while routing a message.

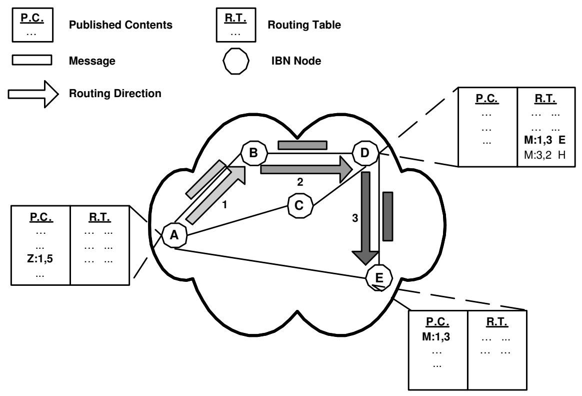

Fig. 2. IBN routing: a message from $Z : 1, 5$ destined to $M : 2, 3$ is routed to $M : 1, 3$

- Different contents are mapped to different nodes based on the algorithm used by the underlying peer-to-peer lookup service. In general, peer-to-peer lookup services assigns keys (content IDs) to nodes uniformly so that no node becomes overloaded and there are techniques to differ the number of keys on each node based on its actual load.
- The proposed implementation is independent of the underlying peer-to-peer lookup services. The implementer can choose from different peer-to-peer lookup services, for example, based on the efficiency of the lookup operation (some peer-to-peer lookup services, such as Pastry [4], takes network locality into their routing algorithm.)

Figure 2 shows an example for the routing in the IBN. A message from $(Z : 1, 5)$ destined to $(M : 2, 3)$ is routed to $(M : 1, 3)$. The message is first routed to D (the node responsible for the mapping of the instances of M) using the underlying peer-to-peer routing mechanism (steps 1 and 2). When the messages reaches node D, it checks its forwarding table and redirects it to the node responsible for the closest instance (Node E).

References

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

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A Scalable Content-Addressable Network," in *Proceedings of ACM SIGCOMM 2001*, 2001.

[3] 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.

[4] 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.

[5] 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.