CS677: Programming Assignment 3
Renaldo: The final saga

Distributed Operating Systems Spring 2003
Department of Computer Science
University of Massachusetts, Amherst, MA 01003 USA
shenoy@cs.umass.edu

1 The problem

This project has two purposes, namely to gain experience with...

- overlay networks
- replication and fault-tolerance

You are encouraged to re-use your work from your second programming assignment as much as possible. In this project, we will focus on the mechanism for sending update messages.

2 Introduction

Upon deploying your server-less Renaldo game concept for a trial period on their networks, Fundingo game corporation found that the game does not scale well with the number of clients. It was found that ALMC from each peer flooded the network. In the extra credit section of programming assignment 2, we alluded to an overlay network (fig 1) as being a cheaper alternative to ALMC for broadcasting requests. In this assignment, we examine the overlay network concept more closely. Application-level multicast does not scale with the number of peers. In a distributed system, we have a physical network which is modeled as a graph.

In this graph, a node represents a host on which a peer runs and an edge represents the physical interconnect between nodes. This graph is sometimes referred to as a host graph (fig 2).

Among the peers we maintain "logical" connections through an RPC mechanism. This is modeled by a graph where nodes are peers and edges are logical

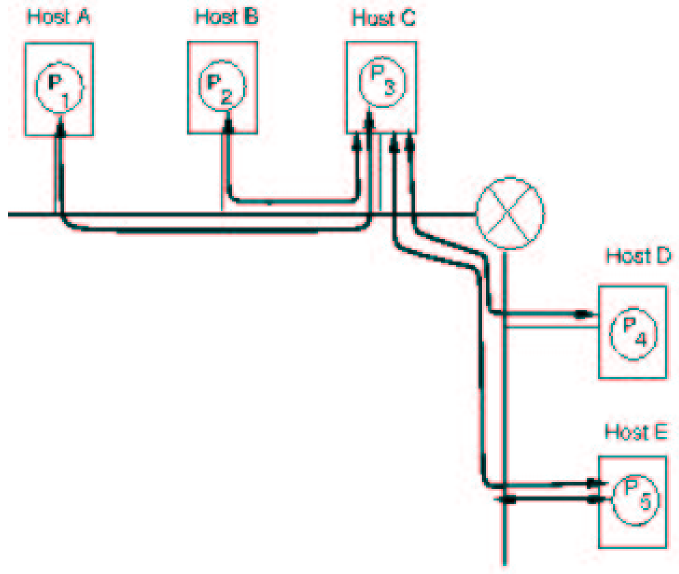

Figure 1: Overlay network

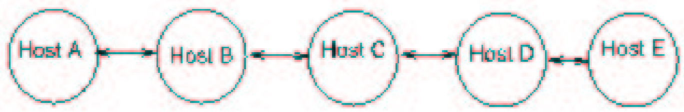

Figure 2: Host graph

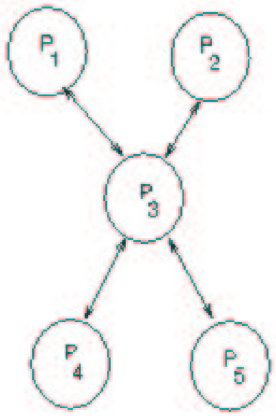

Figure 3: Task graph

connections. This is sometimes called a task graph (fig 3). An overlay network is a task graph built upon or *embedded* in a host graph.

3 Overlay Details

By the ALMC approach, each node maintains a list of peers on the network and sends a unicast message to each peer in the game. Rather than send $n-1$ unicast messages, we organize nodes in a connected graph that is used for performing the multicast. By this approach, a message is forwarded along the task graph and eventually distributed to all nodes. Because the graph is connected, the message reaches every node. Because a node only sends unicast messages to its neighbors, message complexity is reduced.

Again, as we have seen is the case in distributed systems, this added feature comes at a cost. We must somehow manage the graph topology for the overlay network to accomodate nodes which join the game as well as nodes which leave the game. To do this, we introduce a topology manager. A topology manager is a service whose responsibility is to manage the construction of the overlay network. The topology manager implements a graph construction algorithm. When a node joins the game, it is inserted in the overlay network by the topology manager. Likewise, when a node leaves the game, the topology manager manages updates to the overlay network. The topology manager constructs the overlay network by propagating updates to peers informing them of who is adjacent (neighbor) in the overlay graph. It is assumed that the topology manager is available on the network before the game runs. In order to maintain fault tolerance, the topology manager is a replicated service. To simplify things a little bit, the topology manager will maintain a main copy to which reads and writes are issued. The replica copy will service only read requests. In the event that the main copy fails, the replica copy will resume servicing requests.

4 Overlay distribution

Rather than ALMC-ing messages, the ALMC client and service implemented by each peer will be changed to implement an overlay client and service. Requests in the overlay network take two forms. The first is a flooding mode.

In flooding mode (fig 4), a client sends a request by issuing a unicast message to each neighboring node. Upon receipt of a request message, a node forwards it to each of its neighbors after appending its node identifier. If a message arrives at the node for which it is intended, it sends a response message along the reverse path described by the accumulated node ids. Once the response message is received back at the sender, the path to the sender is cached. The next time a message is intended for the same recipient, the cached path is used instead of flooding. Thus, messages are sent along the overlay network by flooding or by cached path. When a cached path is not known, flooding is used to send messages. During flooding, a routes along the overlay network are

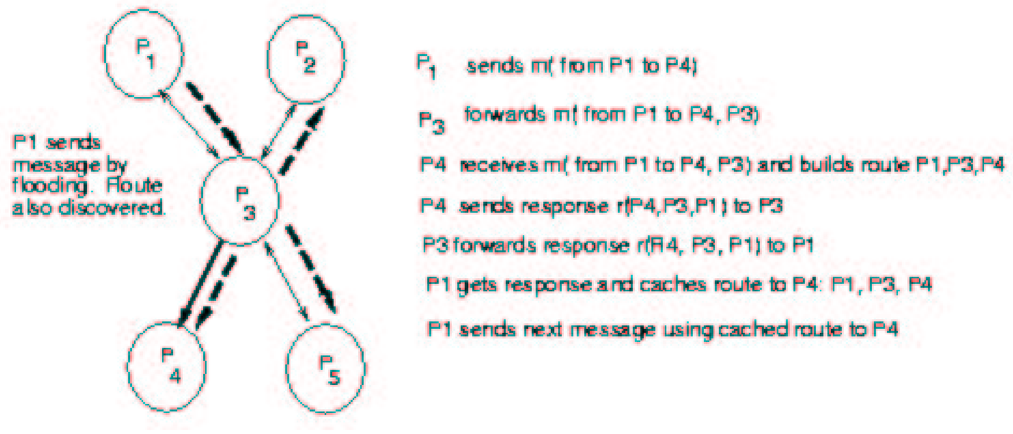
P1 sends message by flooding. Route also discovered.
P1
P2
P3
P4
P5
P1 sends m(from P1 to P4)
P3 forwards m(from P1 to P4, P3)
P4 receives m(from P1 to P4, P3) and builds route P1,P3,P4
P4 sends response r(P4,P3,P1) to P3
P3 forwards response r(P4, P3, P1) to P1
P1 gets response and caches route to P4: P1, P3, P4
P1 sends next message using cached route to P4

Figure 4: Flooding and route caching

discovered by piggybacking an accumulated path on the request. A discovered route is piggybacked on the response message. Discovered routes are cached at the sender. Flooding is a way to perform route discovery in the overlay network. A number of events can trigger route discovery. When a peer sends a request, it sets a timer. If this timer expires before a response arrives, the peer consults the topology manager for a neighbor update and reverts to flooding in an attempt to discover a quicker path.

When the topology manager inserts a new node in the overlay (fig 5), it informs affected nodes with an updated neighbor list. This update causes the affected nodes to revert to flooding.

5 Topology manager

The topology manager maintains an overlay graph structure and an algorithm for inserting and removing nodes which join and leave the network. The reads on the topology manager are performed when someone requests a list of their neighbors. Writes on the topology manager are performed when someone joins the game. Requests to read or write to the topology manager are to be performed using multicast. A read response is made by the topology manager in the form of a unicast message. A write response is made by unicasting an acknoldegement. Writes are serviced only by the main copy. Reads are serviced either by the main copy or the replica. There are two ways to keep the replica consistent with the main copy. By the push approach, writes to the main copy causes it to send an invalidate message to the replica. The replica then invalidates its copy of the data. If a multicast request to read arrives at the replica for a piece of stale data, it ignores the request. Associated with each piece of data stored on the replica is an expiration date. This time lists the duration after which the replica should request the data from the main copy. By the pull

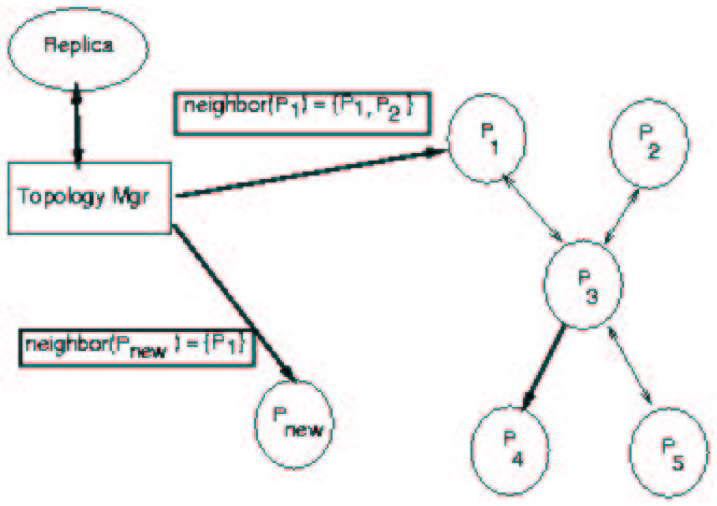

Figure 5: Topology manager updates

approach, the replica server maintains a version number for each piece of data. The replica server periodically polls the main copy by comparing the version number for each piece of data. If the server's version is more up to date, the replica marks its copy as invalid. Again, the data item is associated with an expiration date which governs when the replica downloads a new copy. The data kept in the main copy and replica is an adjacency list. For each node id, its list of neighbors is kept. Data items kept in the main and replica copy are on a granularity of neighbor list. The neighbors for a single node consitutes a data item. The main copy will periodically send "heartbeat" messages to the replica copy. If the replica copy does not receive a heartbeat message, it assumes the role of primary and begins answering write requests in addition to read requests.

6 The assignment

For this assignment, you are to implement...

- topology manager service
- push or pull replication scheme
- overlay network message sending (flooding and cached route)
- heartbeat message and failover

In this assignment, you may simplify the game by reducing the grid to 25×25. You are to test your overlay network with a non-trivial (not a straight line) topology of your choosing with as many peers as necessary to demonstrate the system works. You are to demonstrate a single failover to the replica copy. It is assumed that the primary copy of the topology manager is taken down by killing its server process. You also have leeway about the setting for the heartbeat timer, the expiration date of topology manger records, and the send timer for the overlay network. You are encouraged to be creative in this assignment. If you would like to implement other mechanisms for maintaining consistency between the replica and the primary, you are welcomed. Please make sure to document your work clearly.

7 Comments

As this is the final programming assignment, please include your email address in your project assignment so that the TA may contact you if there are questions about your project submission. Distributed systems is a rich and intersting field where one trades off simplicity, flexibility, scalability against performance. With the first programming assignment, we saw a centralized solution and considered how locking granularity affected performance. In the second assignment, we took a distributed peer approach. Here we saw the kind of issues concerning many aspects of distributed consensus. Since we reach consensus by exchanging messages, we trade off message complexity against consistency. By spending more in message complexity, we saw performance slow because a process has to wait for ACKs to arrive in a totally ordered multicast. By spending less in message complexity, we saw that inconsistent messages may be delivered at a peer causing it to rollback, thus decreasing goodput. Here we try to reduce message complexity using an overlay network. This introduces issues of its own. As a distributed systems person, you will be faced with trade-offs. Depending on your philosophy, you may choose to err on the side of message complexity or goodput or having a fixed number of points of failure (less flexibility). The important part is trying to characterize and understand these issues and trade-offs. We hope that the Renaldo series have helped you to gain an appreciation for these issues.