Token Ring Project

Benjamin Linam and Ali Al-Senan

COP4635 Systems and Networks 2

Protocol Document

As this program requires UDP communication, there are no real connections established between any two users; as such, any “connections” are done by sending and receiving messages. Once the *bbserver* program is executed, it accepts incoming communication from waiting peers. The server then extracts the IP address and port number from each peer, which it then forwards to each peer’s server-designated neighbor to create a ring. Once the server sends each peer’s IP address and port number, it sends the total number of communicating peers so each peer knows how big the ring is.

Once the ring is created, an initiator peer is designated and the server shuts down. A token that enables secure access to a public bulletin board is passed around the ring of communicating peers until any peer requires access to the file. Threading is utilized to ensure access to any peer without the restriction of waiting for another peer to release the token. The thread passes the token and a message to each peer updating the bulletin board file as well as providing unrestricted access.

There are four possible options that a peer can select at any time to be executed the next time it gains access to the token. The write command allows a peer to publish a message to the bulletin board. The read command displays a message for a given sequence number. The list command displays the range of sequence numbers posted to the board. Lastly, the exit command allows a peer to exit by adjusting the ring to skip the user who is exiting.

The ring also allows other users to connect after the server makes the initial connection by providing the IP address and port number for any peer already in the ring. The new peer's IP address and port number are passed around the ring until it reaches the old peer's previous neighbor, who is then redirected to pass the token along to the new peer. As the ring only passes the token and message in one direction, the old peer's previous neighbor is located by use of a global countdown. Each peer keeps a running tally of the total number of peers anytime a peer exits or joins the ring, and once a new peer requests to join the ring, a countdown occurs from the total number of peers to zero to find the correct peer. That peer's next neighbor is updated to be the new peer while the new peer's next neighbor is set to the original requested peer.

When a peer chooses to leave, a message is passed along the ring to let each peer know that the ring is decreasing and size and that the exiting peer is to be skipped during the next token rotation. The exiting peer's previous neighbor's next neighbor is updated to the exiting peer's next neighbor as to prevent loss of the token.

Communication between server and connecting peers:

> **Peer**->**Server**: Connecting
>
> **Server**->**Peer**: Neighbor's IP#: Neighbor's Port#:0/1(whether it has been designated to get initial token): # of Peers

Communication between peers in token ring:

> **Peer**->**Peer**: Entire file passed as text | 0:0:0 (designating that no peer is leaving or exiting)/Neighbor's IP#:Neighbor's Port#:Countdown(designating that a peer is leaving or joining with countdown < 0 if a peer is joining or countdown > 0 if a peer is leaving)

Communication between peer requesting to join and peer in ring

New Peer -> **Peer**: &New Peer’s IP#: New Peer’s Port#

Peer -> **New Peer**: Neighbor’s IP#: Neighbor’s Port#: 0: # of peer’s in ring