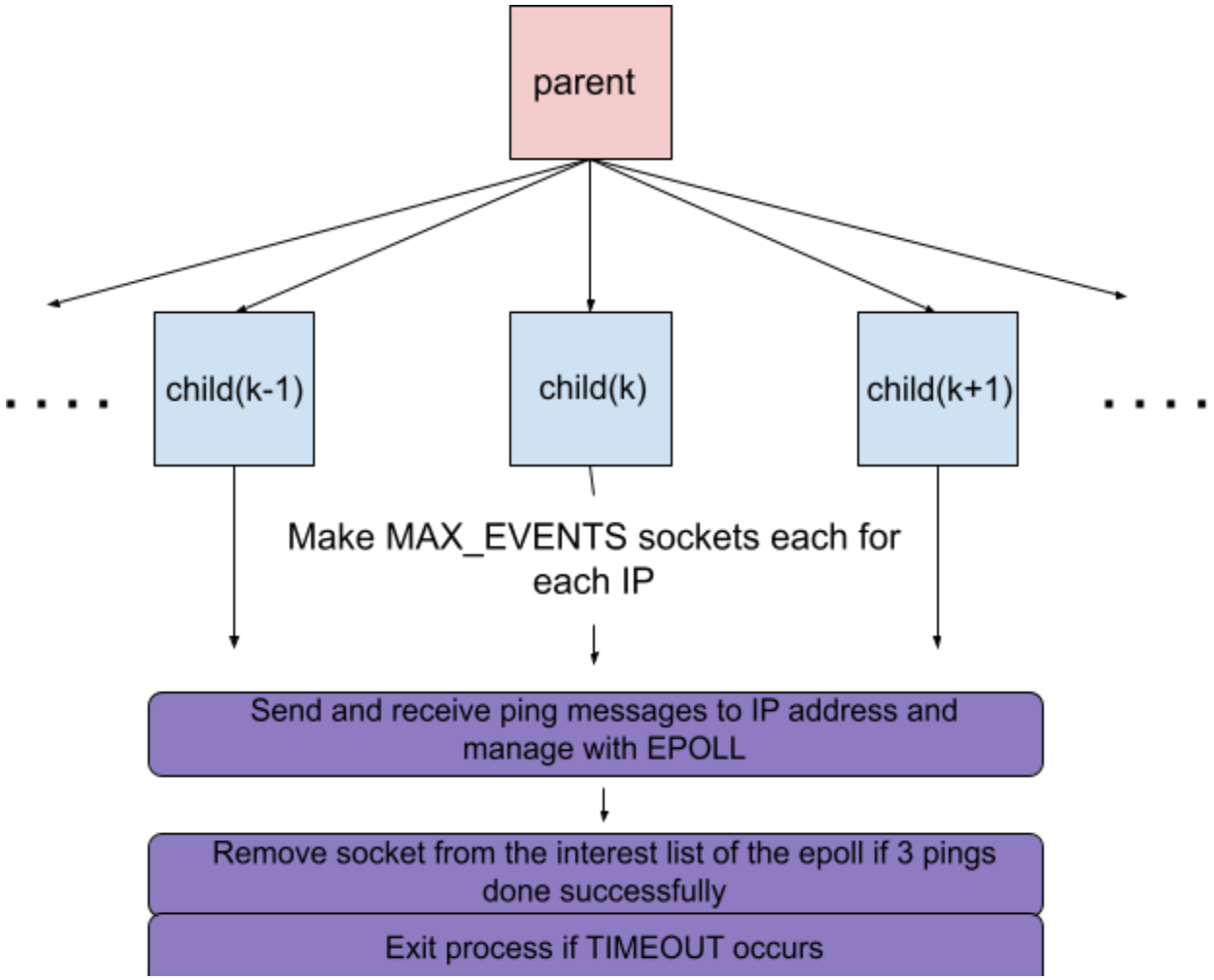
PING Program

Design



Walk through of the design and execution flow

Node is a data structure of all the necessary information we require to ping a particular IP host.

- The main process or the parent process reads the input file and populates the nodes array, until the array is full or the file has reached its end.
- The parent creates a child process, which further populates the nodes array and creates sockets for each node, and assigns a socket to each of them.
- We then add the sockets to the epoll interest list and then continue sending the host ping messages until we successfully receive three replies.

- As soon as we get three successful replies, we remove the node socket from the interest list by closing it.
- When all the sockets are closed for a particular process, we exit from the process.
- Another reason for termination can be the expiry of a user given MAX_TIME value on the command line.

Reason for MAX_TIME (Timeout value)

As there was no way to implement a unique timeout after sending each message, it is possible that the receive call may never terminate. This will occur when the sending call sends an ICMP message and the receiving call is expecting a response but the server never responds. Note that the sender call is not guaranteed to terminate as well. Consider the scenario when the sender sends the first ICMP message for an IP successfully and the receiver does not receive a response for that IP. In that case, the sender will be expecting the receiver to receive a response and add the IP back to the job queue for sending the second ICMP message but since the server did not respond, this never happens.
Therefore a hard timeout value is required.

Features implemented

- Handles both IPv4 and IPv6 addresses.
- Uses ICMP IPv4 and ICMPV6 IPv6 to calculate RTT values.
- Uses epoll with multi-forking for high throughput.

How to execute

1. make
2. sudo ./run [path to input file] [timeout limit in seconds]

Both the arguments are necessary for the execution of the program.
