Questions:

1. **How did you resolve the domain name?**

    We used the 'getaddrinfo' function, which was part of the POSIX networking api. It takes a human-readable hostname like google.com and translates it into one or more usable address structures.

2. **Built-in Ping vs Our Solution:**
    a. **Feature Completeness:** The system ping supports many options out of the box: packet size, number of requests, timeouts, and statistics aggregation. My implementation is a minimal version that only handles the basic echo request/reply exchange with RTT calculation.
    b. **Privileges:** Both rely on raw sockets, but the system ping often uses setuid root or special Linux capabilities (CAP_NET_RAW) so ordinary users can run it. My program requires sudo because it directly opens a raw socket.
    c. **Robustness:** The built-in tool handles more edge cases like fragmentation and error messages (e.g., "Destination unreachable"). My version focuses only on the basic success path.
    d. **Portability:** The system ping handles IPv6 natively (using ICMPv6). My code only works with IPv4 because I explicitly check for AF_INET.

A few optimizations could improve speed and efficiency:

- **Reduce Per-Packet Overhead:** Instead of sending and receiving one packet at a time, I could batch multiple sends and then process multiple replies in one cycle.
- **Shorten Timeouts:** I use a fixed one-second timeout. Lowering this or making it adaptive based on network latency would reduce waiting time.
- **Parallelization:** Instead of sending one packet per second, I could allow multiple requests to be in flight simultaneously and match replies by sequence number.
- **Checksum Optimization:** My checksum algorithm is straightforward but not optimized. A more efficient algorithm could speed up packet creation.