A53280596 Yawen Zhao

# CSE224 HW1

1. Just let all the motes know its temperature and their neighbors temperature. As we are getting the average temperature, we first need to divide all their temperatures by 16. Then from mote 1, send its temperature to its random neighbor. The following neighbor will get its temperature and add it up and send the sum to its next neighbor. Keep doing this until all mote has been visited . The last visited mote can get the average temperature of the furnace.

2. For all the 4 situations, the propagation delay is RTT/2 = 25ms. The result of 4 different situations are as following:
    1) 2 * 50(ms) + 1000 * 1024 * 8 / 1500000(s) + 25 ms = 5.586s
    2) 1000 * 50(ms) + 2 * 50(ms) + 999 * 1024 * 8 / 1500000(s) + 25(ms)= 55.536ms
    3) 50 * 49 + 2 * 50 + 25 (ms) = 2.575s
    4) Suppose $2^0 + 2^1 + \cdots + 2^k \approx 1000$, which is $2^{k+1} - 1 \approx 1000$, we can get k = 9. Then the time will cost equal to $(9 - 0) * 50 + 2 * 50 + 25$ms=0.575s

3. (omitted)

4. The details of this problem are answered as following:
    1) The title of this literature is:
       Beowulf
       An Anglo-Saxon Epic Poem, Translated From The Heyne-Socin
       Text by Lesslie Hall
    2) The keyword is PID: A53280596 Checksum: 231
    3) Attached at the end of this PDF

## Appendix

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>

#include <arpa/inet.h>

#define PORT "5555" // the port client will be connecting to

#define BUFSIZE 100 // max number of bytes we can get at once

// get sockaddr, IPv4 or IPv6:
void *get_in_addr(struct sockaddr *sa)
{
    if (sa->sa_family == AF_INET) {
        return &(((struct sockaddr_in*)sa)->sin_addr);
    }

    return &(((struct sockaddr_in6*)sa)->sin6_addr);
}

void DieWithUserMessage(const char *msg, const char *detail) {
  fputs(msg, stderr);
  fputs(": ", stderr);
  fputs(detail, stderr);
  fputc('\n', stderr);
  exit(1);
}

void DieWithSystemMessage(const char *msg) {
  perror(msg);
  exit(1);
}


int main(int argc, char *argv[])
{
    int sockfd, numbytes;
    char buf[1024];
    struct addrinfo hints, *servinfo, *p;
    int rv;
    char s[INET6_ADDRSTRLEN];
```

```
if (argc != 2) {
    fprintf(stderr,"usage: client hostname\n");
    exit(1);
}

memset(&hints, 0, sizeof hints);
hints.ai_family = AF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;

if ((rv = getaddrinfo(argv[1], PORT, &hints, &servinfo)) != 0) {
    fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(rv));
    return 1;
}

// loop through all the results and connect to the first we can
for(p = servinfo; p != NULL; p = p->ai_next) {
    if ((sockfd = socket(p->ai_family, p->ai_socktype,
            p->ai_protocol)) == -1) {
        perror("client: socket");
        continue;
    }

    if (connect(sockfd, p->ai_addr, p->ai_addrlen) == -1) {
        perror("client: connect");
        close(sockfd);
        continue;
    }

    break;
}

if (p == NULL) {
    fprintf(stderr, "client: failed to connect\n");
    return 2;
}

inet_ntop(p->ai_family, get_in_addr((struct sockaddr *)p->ai_addr),
        s, sizeof s);
printf("client: connecting to %s\n", s);

freeaddrinfo(servinfo); // all done with this structure

socklen_t client_len = sizeof(get_in_addr((struct sockaddr *)p->ai_addr));

const char * message = "A53280596\r\n";
int ret = send(sockfd, message, strlen(message), 0);
if (ret != strlen(message)) {
    DieWithSystemMessage("send() failed");
```

```
    }

    // Receive the response
    ssize_t numBytes;
    do {
        char buffer[BUFSIZE]; // I/O buffer
        numBytes = recv(sockfd, buffer, BUFSIZE - 1, 0);
        if (numBytes < 0)
            DieWithSystemMessage("recv() failed");
        else if (numBytes == 0)
            break;
        buffer[numBytes] = '\0';    // Terminate the string!
        fputs(buffer, stdout);      // Print the echo buffer
    } while (numBytes > 0);

    fputc('\n', stdout); // Print a final linefeed

    close(sockfd);
    fprintf(stderr, "Closed the socket!\n");

    exit(0);

    return 0;
}
```