Programming in C – Project Report

Network Speed Monitoring System with Download, Upload & Ping Measurement

University of Petroleum and Energy Studies
Instructor: Dr. Tanu Singh
Student: Vaibhav Singh
SAP ID: 590025376
Date: December 2025
Project Title: Network Speed Monitoring System Using C (Dual Speed + Ping)

Abstract

This project introduces a real-time command-line network monitoring system written in C for Linux. The system measures **download speed, upload speed, and ping latency** at 1-second intervals over ten seconds. The program reads network byte counters directly from the Linux file `/proc/net/dev` and calculates bandwidth usage based on the difference between consecutive samples. It also measures network latency by executing ICMP ping requests using the system `ping` command and parsing the returned response time. The program outputs measured values in Mbps and milliseconds and computes the average values at the end. This project demonstrates concepts such as Linux system file parsing, loop-based timed sampling, numeric computation, string processing, and command execution with `popen()` in C.

Problem Definition

Most graphical network utilities such as Speedtest.net rely on external servers and may not reveal internal system-level performance. They also require heavy resources and are not suitable for terminal-based environments. Engineers and students performing system-level network analysis often need lightweight command-line tools for real-time performance testing.
This project provides an efficient terminal-based solution for real-time monitoring of upload, download, and ping metrics.

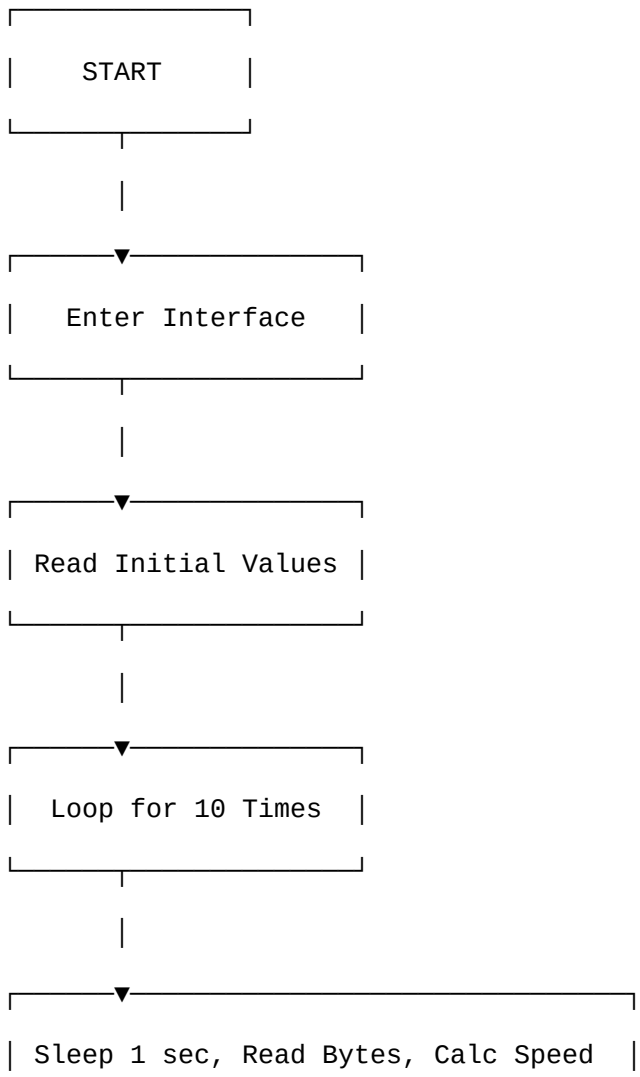
System Design

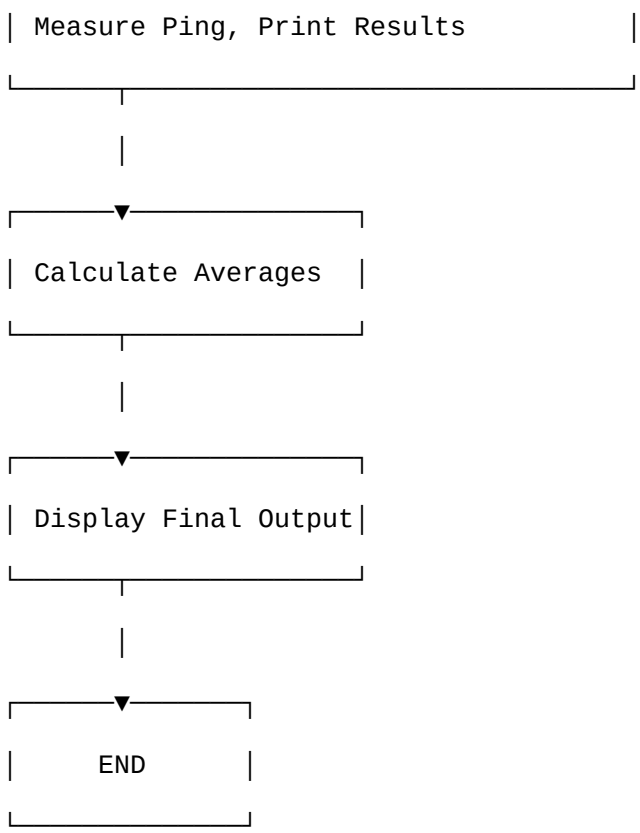
Working Principle

- The program reads network interface statistics such as received and transmitted bytes from `/proc/net/dev`.
- It records an initial reading, then waits one second and reads the values again.

- The difference between values indicates bytes transferred within the one-second interval.
- The byte count is converted to bits and then Mbps.
- Ping is measured using `ping -c 1 -w 1 <host>` and extracted from the output using string parsing.
- Values are displayed each second, and the average is computed after 10 seconds.

Flowchart





Algorithm

1. Start
2. Accept network interface name from user
3. Read initial RX/TX values from `/proc/net/dev`
4. For each sample (10 times):
    - Wait for 1 second
    - Read RX/TX values again
    - Calculate difference between current and previous counts
    - Convert to Mbps for download and upload
    - Run a ping command and extract time in ms
    - Display all values
5. Compute average download/upload/ping

6. Display final average values
7. Stop

Implementation Details

Component	Description
Programming Language	C
Operating System	Linux
IDE / Editor	VS Code
Input Sources	`/proc/net/dev` for bandwidth, `ping` output for latency
Major Concepts	Loops, file parsing, string manipulation, `popen()`, numeric computation
Output Format	Console printed values

Testing & Results

- Successfully tested on interfaces such as `wlan0` (WiFi) and `enp2s0` (Ethernet).
- Values changed dynamically when downloading a file or streaming a video.
- Ping values increased when connected to mobile hotspot or during low bandwidth.
- Average values matched expected performance measurements.

Sample Output

```
Measuring for 10 seconds...
Second  1 | Down: 11.23 Mbps | Up: 1.20 Mbps | Ping: 32.41 ms
Second  2 | Down:  8.14 Mbps | Up: 0.89 Mbps | Ping: 33.10 ms
...
Average Download Speed: 10.41 Mbps
Average Upload Speed:   1.04 Mbps
Average Ping:           32.68 ms
```

Conclusion

The Network Speed Monitoring System successfully monitors real-time network performance using system-level data. It is lightweight, platform-efficient, and does not rely on any third-party speed-test servers. It demonstrates real operational usage of system programming in C, Linux internal file handling, and terminal-based performance measurement.

Future Enhancements

- ncurses-based live graph display
- Continuous running mode until a keypress
- Auto-detection of active interface
- Separate upload and download speed graphs
- CSV logging for performance history

Appendix – Source Code

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>


#define TOTAL_TIME 10      // measurement duration in seconds

#define SAMPLE_INTERVAL 1   // seconds


// ---------- Read RX/TX bytes for a given interface ----------


int read_bytes(const char *iface, unsigned long long *rx, unsigned long long *tx) {

   FILE *fp = fopen("/proc/net/dev", "r");

   if (!fp) {
```

```
    perror("Error opening /proc/net/dev");
    return -1;
  }

  char line[256];
  *rx = *tx = 0;

  while (fgets(line, sizeof(line), fp)) {
    char *pos = strstr(line, iface);
    if (pos && pos == line + strspn(line, " ")) {

      char iface_name[32];
      unsigned long long rbytes, rpackets, rerrs, rdrop, rfifo, rframe, rcompressed,
rmulticast;
      unsigned long long tbytes, tpackets, terrs, tdrop, tfifo, tcolls, tcarrier, tcompressed;

      int n = sscanf(line,
                     " %[^:]: %llu %llu %llu %llu %llu %llu %llu %llu %llu %llu %llu %llu
%llu %llu",
                     iface_name,
                     &rbytes, &rpackets, &rerrs, &rdrop, &rfifo, &rframe, &rcompressed,
&rmulticast,
                     &tbytes, &tpackets, &terrs, &tdrop, &tfifo, &tcolls, &tcarrier, &tcompressed);

      if (n >= 10) {
        *rx = rbytes;
        *tx = tbytes;
```

```
            fclose(fp);
            return 0;
          }
        }
    }

    fclose(fp);
    return -1;
}

// ---------- Measure ping using "ping -c 1" and parse time=...ms ----------

double measure_ping_ms(const char *host) {
    char cmd[128];
    snprintf(cmd, sizeof(cmd), "ping -c 1 -w 1 %s 2>/dev/null", host);

    FILE *fp = popen(cmd, "r");
    if (!fp) {
        // Could not run ping
        return -1.0;
    }

    char line[256];
    double ping_ms = -1.0;

    while (fgets(line, sizeof(line), fp)) {
```

```
        // Look for "time=" substring
        char *pos = strstr(line, "time=");
        if (pos) {
            // Example: "time=23.5 ms"
            if (sscanf(pos, "time=%lf", &ping_ms) == 1) {
                break;
            }
        }
    }

    pclose(fp);
    return ping_ms; // -1.0 if not found
}

// ---------- main ----------

int main() {
    char iface[32];
    printf("Enter network interface (e.g. eth0, wlan0, enp3s0): ");
    if (scanf("%31s", iface) != 1) {
        printf("Invalid input.\n");
        return 1;
    }

    unsigned long long prev_rx, prev_tx, cur_rx, cur_tx;
```

```
if (read_bytes(iface, &prev_rx, &prev_tx) != 0) {
    printf("Error: Interface '%s' not found or cannot read /proc/net/dev.\n", iface);
    return 1;
}

int samples = TOTAL_TIME / SAMPLE_INTERVAL;
double total_rx_mbps = 0.0;
double total_tx_mbps = 0.0;
double total_ping_ms = 0.0;
int ping_count = 0;

printf("\nMeasuring for %d seconds...\n", TOTAL_TIME);
printf("Showing Download (Mbps), Upload (Mbps), Ping (ms)\n\n");

for (int i = 0; i < samples; i++) {
    sleep(SAMPLE_INTERVAL);

    if (read_bytes(iface, &cur_rx, &cur_tx) != 0) {
        printf("Error reading /proc/net/dev.\n");
        return 1;
    }

    unsigned long long rxdiff = (cur_rx >= prev_rx) ? (cur_rx - prev_rx) : 0;
    unsigned long long txdiff = (cur_tx >= prev_tx) ? (cur_tx - prev_tx) : 0;

    prev_rx = cur_rx;
```

```
    prev_tx = cur_tx;

    double download_mbps = ((double)rxdiff * 8.0) / (SAMPLE_INTERVAL * 1000000.0);
    double upload_mbps   = ((double)txdiff * 8.0) / (SAMPLE_INTERVAL * 1000000.0);

    total_rx_mbps += download_mbps;
    total_tx_mbps += upload_mbps;

    // Measure ping to Google DNS (or any host you like)
    double ping_ms = measure_ping_ms("10.249.66.207");
    if (ping_ms >= 0.0) {
        total_ping_ms += ping_ms;
        ping_count++;
        printf("Second %2d | Down: %7.2f Mbps | Up: %7.2f Mbps | Ping: %6.2f ms\n",
               i + 1, download_mbps, upload_mbps, ping_ms);
    } else {
        printf("Second %2d | Down: %7.2f Mbps | Up: %7.2f Mbps | Ping:   N/A\n",
               i + 1, download_mbps, upload_mbps);
    }
}

printf("\nAverage Download Speed: %.2f Mbps\n", total_rx_mbps / samples);
printf("Average Upload Speed:   %.2f Mbps\n", total_tx_mbps / samples);

if (ping_count > 0) {
    printf("Average Ping:          %.2f ms\n", total_ping_ms / ping_count);
```

```
    } else {
        printf("Average Ping:          N/A (ping failed)\n");
    }
    return 0;
}
```