ECE 445

SENIOR DESIGN LABORATORY

PROJECT PROPOSAL

Arduino-Powered Network Flow Visualization Toolbox

Team #17

BOLIN ZHANG
(bolinz3@illinois.edu)
JIAHAO FANG
(jiahaof3@illinois.edu)
YIYANG HUANG
(yiyangh5@illinois.edu)
ZIYUAN CHEN
(ziyuanc3@illinois.edu)

TA: TBD

March 6, 2024

Contents

List of Figures

1 Introduction

1.1 Problem

Many real-world systems involve flows over networks. Logistic systems, transportation networks, and the Internet are all carefully designed to meet the capacity and cost requirements. However, in algorithm courses, flow optimization problems can be hard to imagine. Students often struggle to quantitatively predict how each twerk in the constraints will affect the optimal solution *using only intuition*.

1.2 Solution

Our team aims to build a *modular, reconfigurable* hardware emulator to visualize network flows under capacity constraints on links. Each node can be configured as a source, a sink, or a "transfer station" that holds zero flux. Solutions will be computed using an embedded Arduino microcontroller. This toolset will provide an intuitive visual aid and facilitate the understanding of flow algorithms in a classroom setting, especially where the network in discussion is inherently *dynamic* (e.g., routing packets in the Internet).

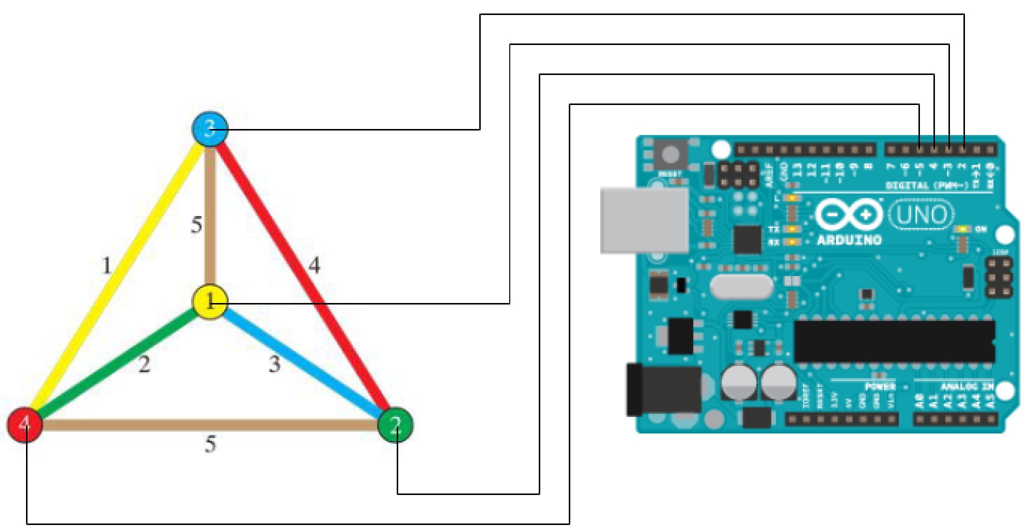

Figure 1: Pictorial representation of the physical model.

1.3 High-Level Requirements

- The physical model should be modular, i.e., each node has a number of "slots" reserved for installing new links. We aim to serve *4 to 6* fully connected nodes.
- The Arduino software should communicate with all nodes and pipes and update the flows in real-time (*within 500ms*) in response to changes in setup.
- The algorithm should handle and report edge cases such as a network with *zero or multiple* feasible flows.

2 Design

In the **hardware network model**, pipes represent network links and the LEDs within show the maximal capacity and real-time flow of “data packets.” Each node is a PCB board configurable as source, sink, or neither (“transfer station”) with a user-friendly interface such as buttons or switches. We use a *scalable* design where components are easily replaceable to account for network expansion.

The **software flow computer** implements a robust and *lightweight* optimization algorithm that efficiently computes network flows on an embedded Arduino board while considering all constraints (node configurations, link capacities). If time permits, we may also develop a software GUI to display the network alongside the physical model due to limited space (number of LEDs and pins for interconnection) in each node and link.

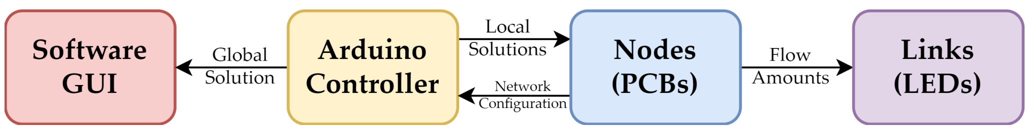

Figure 2: Subsystem organization of top-level design entity.

2.1 Hardware Subsystem

By “hardware”, we are referring to the physical network model with nodes and links, while the Arduino board is considered primarily related to software. Each node is a customized PCB board *with minimal size* that includes

- a 3-mode switch that configures the node as a source, sink, or transfer station,
- two digits of 7-segment display that shows the amount of flow from (for sources), to (for sinks), or through (for transfer stations) the node in the optimal solution,
- at least 3 slots (pins) for installing new links (LED strips), and
- a microcontroller to drive the interface and communicate with the Arduino board.

We considered treating the links as mere LED strips and leave all capacity configurations to the node PCBs, but this appears counterintuitive and would require a complex communication protocol to set up the network topology. Until we come up with a more lightweight solution in the design document, we regard the links as PCBs with

- a knob that sets the link capacity (*tentative*),
- two digits of 7-segment display that indicates the capacity (*tentative*),
- a string of LEDs that shows the actual flow,
- 2 slots (pins) for installing new links (LED strips), and
- a microcontroller that talks with the nodes about its capacity and actual flow.

2.2 Software Subsystem

The emulator has a central Arduino controller that talks to each node (but *not* the links) to display the capacities and actual flow amounts. At the current stage, we plan to employ an Arduino UNO and dedicate one to two I/O pins for each node, though this inherently limits the number of nodes we can support. The microcontroller is expected to

- read the node configurations and link capacities (*from which node?*),
- computes the maximal flow using an optimized Ford-Fulkerson algorithm, and
- updates the flow display on each node and link in real-time.

For the software GUI, we considered an intuitive interface that allows the user to easily configure nodes (3 modes) and links (capacity) while controlling the LED flow display, but such a centralized design would require lots of connections and is less robust and scalable. Therefore, the GUI simply displays the network topology and the maximal flow solution at this point – though we may change our mind in the design document.

3 Ethics and Safety

3.1 Ethical Concerns

To avoid ethical breaches in the development and deployment of our toolbox, we commit to adhering closely to the principles outlined in both the IEEE Code of Ethics [1] and the ACM Code of Ethics [2]. Key considerations include but are not limited to

- Respecting *intellectual honesty* (ACM, Clause 1.5), acknowledging contributions accurately, adopt secure coding practices, and avoiding plagiarism in development.
- Committing to *inclusivity and accessibility* (ACM, Clause 1.4). For example, both the model and the GUI should be designed to be usable by a broad spectrum of individuals and accommodate users with diverse technical backgrounds.
- Supporting *sustainable development* (ACM, Clause 3.4; IEEE, Clause 1). This includes choosing recyclable and sustainable materials for hardware components and designing the embedded electrical system for energy efficiency.
- Mitigating the *risk of overreliance* by positioning our tool as a supplementary, instead of replacement, of traditional educational resources, in compliance with the IEEE's commitment to continuous learning (IEEE, Clause 6).

By fostering an environment of transparency, responsibility, and respect for user rights, we aim to not only comply with professional ethical standards but also contribute positively to the educational and technological communities.

3.2 Safety Concerns

This project involves the use of diodes, microcontrollers, and light bulbs to simulate the network information transmission flow. Recognizing the associated electrical, fire, mechanical, chemical, and operational hazards both in the development and deployment stages, we will abide by the IEEE National Electrical Safety Code [3] through

- Implementing comprehensive safety measures including protection against *electric* shocks (e.g., insulated tools) and *fire* precautions (e.g., circuit breakers, fuses) to prevent overheating and short circuits.
- Securely mounting all PCB board components to ensure *mechanical* robustness.
- Using protective gears during assembly involving batteries and soldering operations, and safely disposing hazardous *chemical* waste.
- Ensuring the safety of users (ACM, Clause 2.9; IEEE, Clause 1) by rigorously testing the system to prevent any *operational* hazards. For instance, the number of small parts in the physical model should be minimized to prevent choking hazards.
- Training in safe handling practices and emergency procedures.

References

[1] IEEE. "IEEE Code of Ethics." (2016), [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html (visited on 03/06/2024).

[2] ACM. "ACM Code of Ethics." (2018), [Online]. Available: https://www.acm.org/code-of-ethics (visited on 03/06/2024).

[3] "2023 National Electrical Safety Code® (NESC®)," *2023 National Electrical Safety Code(R) (NESC(R))*, pp. 1–365, 2022. DOI: 10.1109/IEEESTD.2022.9825487.