Energy Harvesting Node Algorithm for EnHANTs

Caroline E. Schiavo

Research performed at Columbia University's Electrical Engineering Wireless and Mobile Networking Lab

Author Note: This research was designed and guided by Jelena Marašević, a Ph.D. student in the Columbia University Department of Electrical Engineering in the laboratory of Professor Gil Zussman. This research was conducted in Professor Gil Zussman's Wireless and Mobile Networking Lab at the Columbia University Schapiro Center (CEPSR).

Energy Harvesting Node Algorithm for EnHANTs[1]

ABSTRACT

EnHants, Energy Harvesting Active Networked Tags will supersede the use of ultra-low power networks, helping survivors of natural disasters and finding misplaced objects. The energy harvesting is used for multiple nodes to form a network that transmits to one main node or device [1, 4, 13, 14]. Each node has a set algorithm of iterations and decisions, using an initial battery state, a battery capacity, number of time slots, and harvested energy, battery states, and a fixed amount of energy per time slot [1]. This existing algorithm determines how much energy one node should spend overtime, while this research does not involve the entire network of nodes. This implementation was to understand how the battery state evolves over time for a given initial state, the amount of energy that is harvested over time, and the battery capacity, when the device spends a constant input amount of energy *x* per time slot. Software Dev C++ assisted in developing the algorithm and certain libraries are imported to run the code. After the algorithm is determined, it is compiled and run in many test trials, until an effect can be seen by inputting different integer values [2]. It can be observed what is the best energy harvested for each time slot and what are the values of b. One can answer: Why is energy harvesting an advantage to other network strategies? Then, energy harvesting will be an enabler and building block for the Internet of Things.

Categories and Subject Descriptors: C.2.1 [Computer-Communication Networks]: Network Architecture and Design — *Wireless Communication*

General Terms: Algorithms, Measurements, Index

Keywords: Nodes, Energy Harvesting, EnHANTs, Routing, Battery states, Input and output streams, ultra-low power, Internet of Things (IoT)

[1]EnHANTs stands for Energy Harvesting Active Networked Tags, which is a project developed within Columbia University's Electrical Engineering: Wireless and Mobile Networking Lab.

1.INTRODUCTION

The creation of a node algorithm, along with energy harvesting and the expansion of extremely low-power systems will allow for self-powered networked nodes that outweigh the advantages of using low-power Bluetooth, Wi-Fi, and other suitable networks [4, 13, 14]. In order for self-powered nodes to be established, there must be a specific algorithm set for each node to transmit the signal using resource allocation, which assigns the available resources to and from the node [1]. This allocation will change depending on the type of energy available within each node. The algorithm computes the battery levels for a given constant amount of energy spent per time slot.

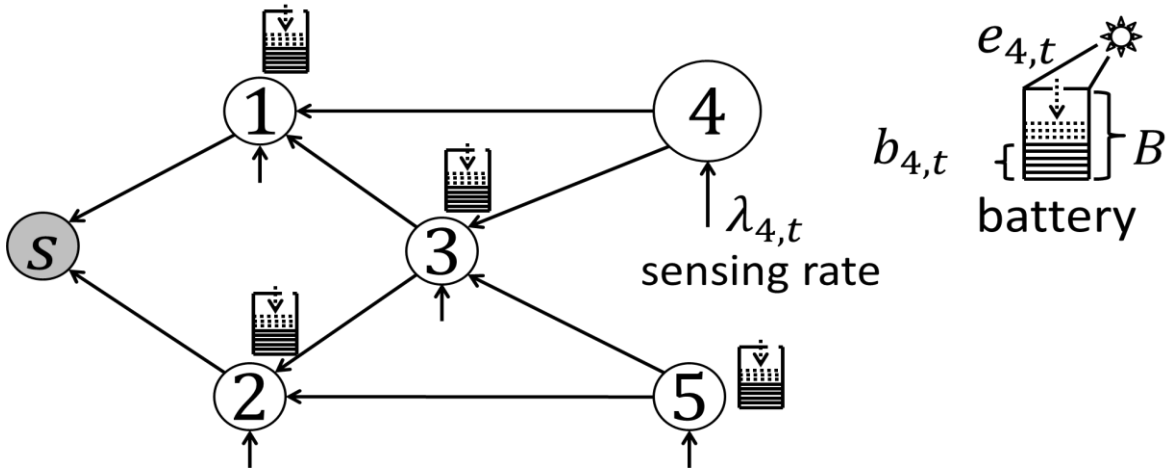


Figure 1. An energy harvesting network with multiple nodes, an endpoint for the data at s, and a battery with given variables [1].

Table 1. Nomenclature [1]

1 node = 1 device	
1 unit of data = 1 unit of energy	
T	Number of time slots
b1	Initial battery state
B	Battery capacity
*e	harvested energy per time slot
*b	Battery states per time slot
X	Fixed amount of energy per time slot, spend b_t
I	Integer for iteration of for loop

An energy harvesting network captures small amounts of energy that would be lost otherwise, where the nodes sense the surroundings and forward the data. The data from the static nodes end at its' destination, *s*, as seen in Figure one over many *T,* time slots. The battery of each node has a certain limit for *B*, the battery capacity, which is related to *b1*, the initial battery level. The energy, **e*, is harvested for time slots, while its companion *x* inputted by the user is a 'fixed amount' of the energy that node spends per time slot. For the purpose of the lab and this paper, there is one node present in the harvesting network communicating with the tags. In regards to the nodes, they have rechargeable batteries that can store small amount of energy. This stored energy increases as the nodes harvest energy and when energy is spent, there is a decrease. For many different energy sources, the amount of energy that is available for harvesting is constantly changing over time, which has been observed through measurements such as traces of indoor and outdoor light [23].

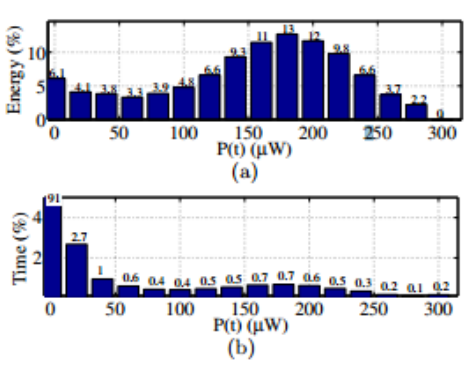


Figure 2. Motion energy harvesting process variability regarding time and energy

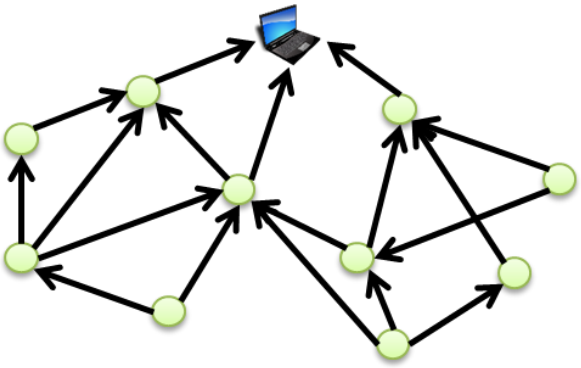


Figure 3. Another simple energy harvesting network where the green dots are the devices trying to reach the endpoint, the laptop.

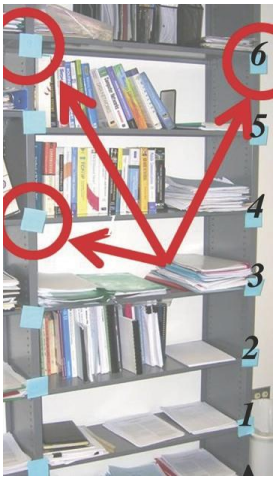


Figure 4. Demonstration and the enabling of EnHANTs as is related to shifts, in layman's terms the solar cells on the bookcase [24].

The paper is organized as follows. Section 2 provides step-by-step instructions on the amount of energy held at each time slot, while Section 3 explains the results of this method. Section 4 concludes the paper, by connecting the meaning of the results to the bigger picture.

The goal of the paper is to address the problem associated with network overload by creating an algorithm for each node to connect to one and other and a specific destination. This problem can be dealt with by using Energy Harvesting Active Networked Tags to assist The Internet of Things, the network of physical objects through the internet, allowing for circumventing of novel applications and methods to

be replaced by a stronger infrastructure. For example, this condensed and advanced communication will allow misplaced small items such as keys to be found, the constant monitoring of devices, and swifter human rescue operations.

2.MATERIALS AND METHODS

The node algorithms required the use of the C++ or Java programming languages. Java was originally used in developing the algorithm for the node. Later, after discussion, the language of C++ was thought to be the best language for integration. The coding was compiled and run in Dev C++[2], a free IDE and compiler for C and C++ programming languages [2]. The requirements of the program required the addition of three libraries cstdlib, iostream, and fstream that assisted the necessity of including the input and output file stream classes. On July 21st, it was determined that cin can be used to input and assign a value to *x*, the fixed amount of energy, after a command to "Enter x" is provided by cout. The input file stream class' variable, fin, allowed the number of time slots, initial battery state, and the batter capacity to be inserted through a text file named "input.txt". Logical thinking was utilized to interpret that the first or initial value of an object is always the index of zero, not the index of one in an array, which allowed the initial battery state to be set equal to the first array. The harvested energy has time slots represented by *T*, so *e* is set equal to the length of the array of *T*, time slots, which is the same as *b*, the time slots for the battery states equal to *T* plus one slots [1-3].

---

[2]C++ plus compiler made by Orwell and downloaded using SourceForge or Bloodshed.

Then, the variable *i* was declared to instantiate the iteration of the for loop and input and print the harvested energy. By July 24th, another for loop iteration was created for more output repeating until the end of the time slots is reached or other factors affect the iteration. This time, the slots for the battery state had an index of i plus one. That index of b equals *x* subtracted from index i of b and e would skip the decision statement and repeat again. Associated, the if condition must be checked to see if b[i+1] is not only greater than the battery capacity, but the probability that it could be equal to the capacity. Later in this process, the output was found for all the values of b, slots for the battery capacity by initializing as zero, incrementing by one, and having the termination value be the number of *T*, time slots available or given for a practice test run. All operations applied at the start of the algorithm need to be terminated so e and b are deleted, while the ifstream's variable fin is closed. The algorithm was compiled and run without any error, while the user inputted different values of x for test trials, representing the differences in the battery and energy. As another trial, the user edited the notepad file named 'input' by switching the number of time slots and varying the initial battery state and battery capacity. These test trials were conducted, for the purpose of visualizing and interpreting the fluctuating data and results

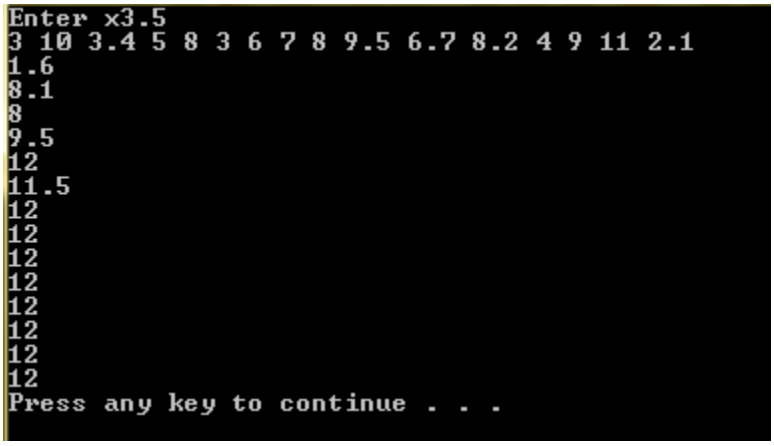
Table 2. C++ Input and Output Library [3]

Cout<<	Standard output stream, object
Cin>>	Standard input stream, object
Fin>>	Input file stream class variable name
Fout<<	Output file stream class variable name
Ofstream fout	Output file stream class
Ifstream fin	Input file stream class

3.RESULTS

The data was collected after compiling and running the program in the command prompt window. The command prompt which is located inC:\javaprogs\Columbia\GSTEM\GSTEM.exe included the Enter x output. The user tested multiple integer values for *x,* until satisfied with the *x* value of 3.5. The same process occurred to allow for fifteen time slots. After testing multiple numbers for the battery capacity, *B*, it was determined that increasing the battery capacity lead to the change in energy harvesting for more time slots, while decreasing the battery closer to the initial battery state left much energy per time slot at the same level as the battery capacity. This can be demonstrated with twelve as the battery capacity; this lower capacity shows only five amounts of energy that are harvested or changed (Figure 4), while with the battery capacity of thirty-five, almost all the fifteen time slots have been harvested with a different amount of energy than the battery capacity (Figure 5). Regardless of the results, all fifteen original battery states or levels for each time slot appeared after running the code (Figure 5 & 6).

Figure 5. Command Prompt, 12 [2]



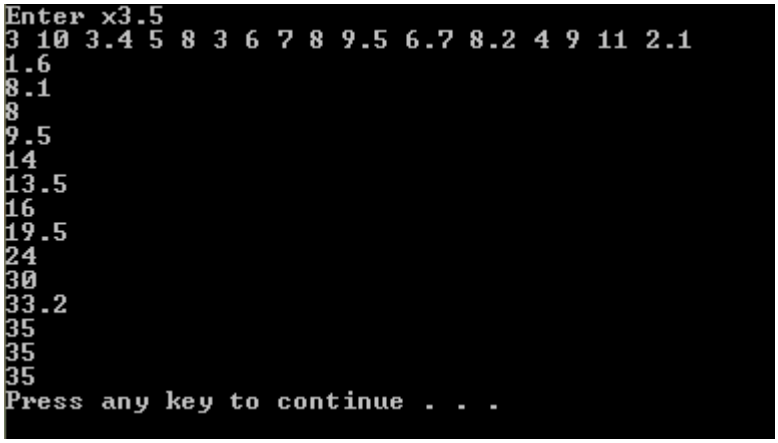


Figure 6. Command Prompt, 35 [2]

4.DISCUSSION

The results conducted are accurate to previous results conducted by others and did not refute or prove this hypothesis. This algorithm, in the future, can be implemented for all the nodes. Then the algorithm can be used to look at the entire network and interdependencies of data rates assigned to the nodes. This research is just one part of the big puzzle of the energy harvesting network representing the potential benefits of the EnHANTS for the Internet of Things.

ACKNOWLEDGEMENTS

Thank you to Associate Professor of Electrical Engineering, Gil Zussman and Ph.D. student, Jelena Marašević from Columbia University. They have been instrumental in my success with theoretical coding work, teaching me the C++ programming language within the environment of Dev C++, and supportive in my adjustment to the anticipated high rigor college work. Thank you to everyone else at the Wireless and Mobile Networking Lab.

LITERATURE CITIED

[1] J. Marašević, C. Stein, and G. Zussman. Max-min fair rate allocation and routing in energy harvesting networks: Algorithmic analysis. CoRR, abs/1406.3671, June 2014.

[2] Orwelldevcpp. Orwell Dev C++. Sourceforge, 2014.

[3] Input/Output Library. Cplusplus.com, 2000-2014.

[4] Energy-Harvesting Active Networked Tags (EnHANTs) Project, Columbia University, http://enhants.ee.columbia.edu.

[5] S. Chen, P. Sinha, N. B. Shroff, C. Joo. Finite-horizon energy allocation and routing scheme in rechargeable sensor networks. IEEE INFOCOM'11, 2011.

[6] B. Bacinoglu and E. Uysal-Biyikoglu. Finite-horizon online transmission rate and power adaptation on a communication link with markovian energy harvesting. CoRR, abs/1305.4558, 2013.

[7] D. Bertsekas and R. Gallager. Data networks (2nd ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[8] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. IEEE/ACM Trans. Netw., 12(4):609-619, 2004.

[9] A. Charny, D. D. Clark, and R. Jain. Congestion control with explicit rate indication. In Proc. IEEE ICC'95, 1995.

[10] S. Chen, P. Sinha, N. Shroff, and C. Joo. A simple asymptotically optimal energy allocation and routing scheme in rechargeable sensor networks. In Proc. IEEE INFOCOM'12, 2012.

[11] M. Gatzianas, L. Georgiadis, and L. Tassiulas. Control of wireless networks with rechargeable batteries. IEEE Trans. Wireless Commun., 9(2):581-593, 2010.

[12] M. Gorlatova, A. Bernstein, and G. Zussman. Performance evaluation of resource allocation policies for energy harvesting devices. In Proc. WiOpt'11, 2011.

[13] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman. Challenge: ultra-low-power energy-harvesting active networked tags (EnHANTs). In Proc. ACM MobiCom'09, 2009.

[14] M. Gorlatova, R. Margolies, J. Sarik, G. Stanje, J. Zhu, B. Vigraham, M. Szczodrak, L. Carloni, P. Kinget, I. Kymissis, and G. Zussman. Energy harvesting active networked tags (EnHANTs): Prototyping and experimentation. Technical Report 2012 07-27, Columbia University, July 2012.

[15] M. Gorlatova, A. Wallwater, and G. Zussman. Networking low-power energy harvesting devices: Measurements and algorithms. IEEE Trans. Mobile Comput., 12(9):1853-1865, 2013.

[16] B. Gurakan, O. Ozel, J. Yang, and S. Ulukus. Energy cooperation in energy harvesting two-way communications. In Proc. IEEE ICC'13, 2013.

[17] R.-S. Liu, K.-W. Fan, Z. Zheng, and P. Sinha. Perpetual and fair data collection for environmental energy harvesting sensor networks. IEEE/ACM Trans. Netw., 19(4):947-960, Aug. 2011.

[18] Z. Mao, C. Koksal, and N. Shroff. Near optimal power and rate control of multi-hop sensor networks with energy replenishment: Basic limitations with finite energy and data storage. IEEE Trans. Autom. Control, 57(4):815-829, 2012.

[19] N. Megiddo. Optimal flows in networks with multiple sources and sinks. Mathematical Programming, 7(1):97-107, 1974.

[20] O. Ozel, K. Tutuncuoglu, J. Yang, S. Ulukus, and A. Yener. Transmission with energy harvesting nodes in fading wireless

channels: Optimal policies. IEEE J. Sel. Areas Commun., 29(8):1732-1743, 2011.
[21] S. Sarkar and L. Tassiulas. Fair allocation of discrete bandwidth layers in multicast networks. In Proc. IEEE INFOCOM'00, 2000.
[22] R. Srivastava and C. Koksal. Basic performance limits and tradeoffs in energy-harvesting sensor nodes with finite data and energy storage. IEEE/ACM Trans. Netw., 21(4):1049-1062, 2013.
[23] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, and G. Zussman, "Movers and shakers: Kinetic energy harvesting for the Internet of things," in *Proc. ACM SIGMETRICS'14*, 2014.
[24] M. Gorlatova, P. Kinget, I. Kymissis, D. Rubenstein, X. Wang, and G. Zussman, "Energy harvesting active networked tags (EnHANTs) for ubiquitous object networking," *IEEE Wireless Communications, Special Issue on the Internet of Things: The Next Big Thing in Communications*, vol. 17, no. 6, pp. 18–25, Dec. 2010.

APPENDICES

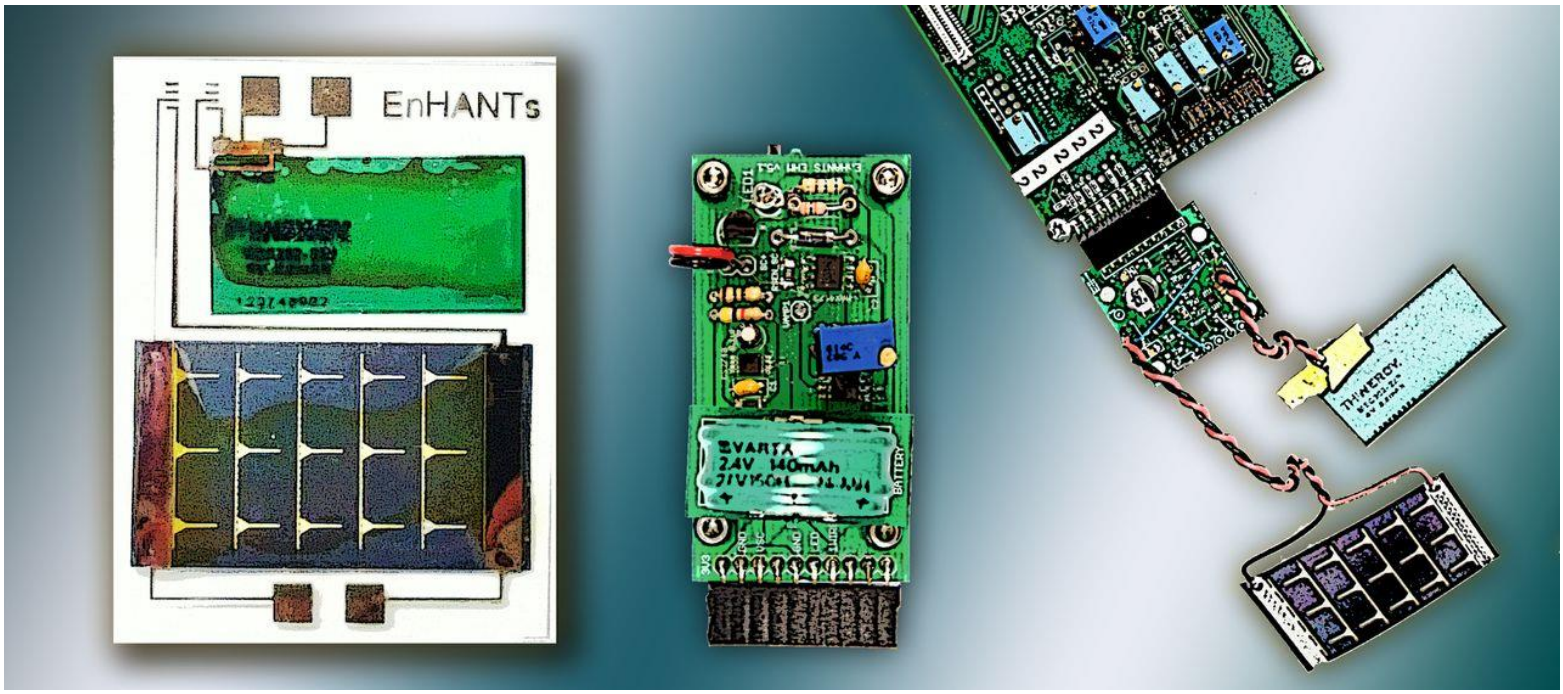


Figure 7. Example of EnHants pieces

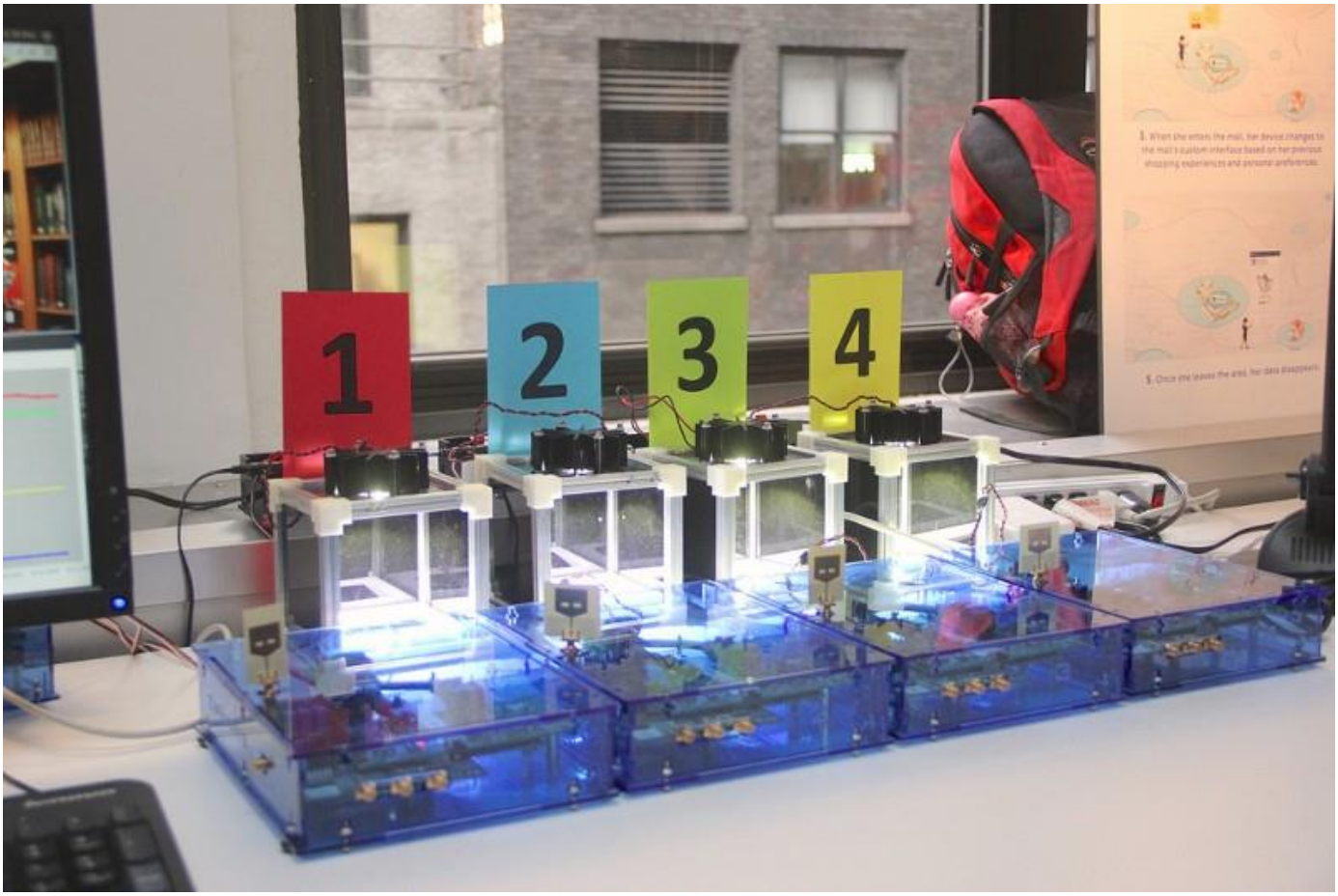


Figure 8. The energy harvesting network with four nodes

Code Examples

```
#include <cstdlib>
#include <iostream>
#include <fstream>

using namespace std;

int main(int argc, char *argv[])
{
   ifstream fin("input.txt");
    int i; // integer i used in for loop
    int T; // Number of time slots
    double b1; // Initial battery state
    double B; // Battery capacity
    double *e; // Harvested energy for slots 1,2,..., T
    double *b; // Battery states for slots 2, 3,..., T+1
    double x; // Fixed amount of energy spent per time slot

    // Before doing this part you need to enter T either from the standard input
    // or from a file
    //At this point, also input
    fin>>T;
    fin>>b1;
    e = new double [T];
    b = new double [T+1];
    b[0] = b1;
    cout<<"Enter x";
    cin>>x;
    fin>>B;
```

```
    for (i = 0; i < T; i++)
    {
        fin>>e[i]; //Inputing and printing harvested energy
        cout<<e[i]<<" ";
    }

    for (i = 0; i < T; i++)
    {
        b[i+1] = b[i] + e[i] - x;

              if (b [i + 1] > B)
               {
                    b[i + 1] = B;
              }
    }

   for (i = 0; i < T; i++) //For loop that outputs all the values of b
    {
      cout << b[i] << endl;
    }
    delete [] e;
    delete [] b;
    fin.close();
    system("PAUSE");
    return EXIT_SUCCESS;
}
```