Teaching Routing with IRV-Tool

Michael Welzl, Muhammad Ali
Institute of Computer Science
University of Innsbruck, Austria

Abstract

The IRV-Tool ("Internet Routing Visualization") is a simple interactive tool designed to give students easy access to the fundamentals of Internet routing. Two major classes of dynamic routing protocols — distance vector and link state — can be studied with this tool; for distance vector routing, we already successfully used it in classes. In this paper, we introduce the IRV-Tool, describe example exercises that we gave to our students and report on student feedback, which was quite positive.

1 Introduction

Routing is an integral part of the Internet. A huge amount of material about routing is available in various books that are used to teach the fundamentals of computer networks. Ideally, students would work on exercises in a live testbed, but this method is often impossible due to obvious difficulties: what if there is a large number of students (e.g. around 100) who need to carry out their work within the same time period? What happens when the deadline for delivering a project approaches — will students have to fight for testbed use time? Also, building a testbed which can accommodate a large number of students is expensive, both in terms of hardware and manpower costs. Thus, in practice, routing exercises are often theoretic in nature and typically solved on a piece of paper.

In order to bridge this gap between theory and practice, and facilitate the learning process, we developed the IRV-Tool ("Internet Routing Visualization"). Effectively, this tool is a network simulator, albeit one that is designed to simulate routing only. It covers two major classes of dynamic routing: distance vector and link state. We implemented most of the main features of distance vector routing and some features of link state. An extension is already underway to include other features of link state routing.

The IRV-Tool is part of our general concept for teaching computer networks: we use a number of graphical interactive tools that are written to accommodate the needs of students as they study the diverse range of topics to be tackled in the course. Since each tool is customized for a single topic, it is possible to make them very easy to handle. Other tools that we are using include the "Congestion Avoidance Visualization Tool (CAVT)" [1] and "Network Simulation By Mouse (NSBM)", a GUI for designing ns-2 scenarios [2]. We are clearly not the only group to follow such an approach — examples from other colleagues include the Java based sniffer tool described in [3], the toolkit described in [4], and the tools in the teaching section of the java.net website.[1]

2 Related Work

Many vendors, like opnet[2], provide propreitery tools for various network solutions for a wide range of services There are, however a few open source general purpose simulation tools available which are widely used for teaching and research purposes. OLSR[3], for instance, is an implementation of the optimized link state routing protocol for mobile ad-hoc networks. Similarly, tools like ns-2[4], omnet++[5] and otter[6] are commonly used to study and teach various aspects of computer networks and the Internet, including routing. However, these tools cover wide areas in computer networks and require a handful amount of coding and scripting expertise. While there is limited amount of time to teach networking principles in the class, most of the time is consumed in learning to use tools like ns-2 or omnet++. This fact is underlined by the *negative* feedback that we obtained from the students when trying to teach them congestion control with ns-2, where they were spending most of their time trying to learn how to use the simulator and not the actual course contents that we wanted to convey. The IRVtool is a custom-made tool specifically meant for the students to learn the detailed workings of both distance vector (along with problems associated with it) and link state routing in a

[1]https://edu-tools-for-teaching.dev.java.net/

[2]http://www.opnet.com/

[3]http://www.olsr.org/

[4]http://www.isi.edu/nsnam/ns/

[5]http://www.xomnetpp.org/

[6]http://www.caida.org/tools/visualization/otter

convenient way. By using the IRVtool, students can observe the real behavior of routing and control the overall environment in a simplified and interactive manner. This way, they can put more time and effort in learning the core of dynamic routing than the semantics of simulators like ns-2 or omnet++.

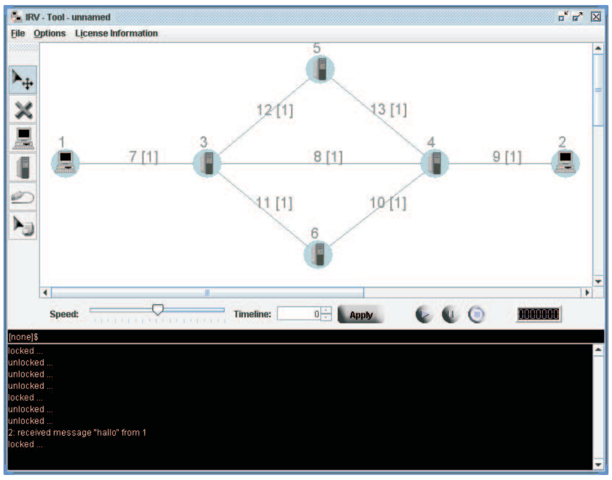
This paper is organized as follows: in the next section, we provide a brief description of the distance vector and link state routing mechanisms. Section 4 introduces our tool. We present three of the tool-based exercises that we gave to our students in section 5, and discuss student feedback in section 6; section 7 concludes the paper.

3 Dynamic Routing

Routing is the act of moving information from a source to a destination following the shortest possible path available. Routing occurs at the network layer of the OSI reference model; it can be static or dynamic. In static routing, any changes in the path are manually performed by an administrator, whereas in dynamic routing alternate paths are automatically selected when links or routers fail. There are two major classes of dynamic routing; *Distance Vector (DV) routing* and *Link State (LS) routing*: [5]

- DV routing, also known as Bellman-Ford algorithm, is a simple routing method where each router maintains a table about the known destinations available to it along with the link to get there, and it sends this information to its neighbors. This way DV routing only keeps information about the neighbors and not the whole network. However, distance vector routing is known to have some serious problems; one of them is “count to infinity”. This is a very common and well known problem in DV routing which is taught to the student in the network course (we teach it using IRVtool).

- LS routing, in contrast to DV, tells every router on the network about its neighbor. The mechanism is based upon a “distributed map”, where each router has a copy of the map that is regularly updated. Hence after convergence of the network, each router has identical copy of the routing table. On the negative side, the scalability of link state routing is limited (more so than in the distance vector routing case), and there are numerous other problems that must be dealt with. It is worth mentioning here that in the current version of the tool, only elementary behavior of link state routing can be observed. Problems and extensions to the algorithm cannot be investigated



```
[none]$
locked ...
unlocked ...
unlocked ...
unlocked ...
locked ...
unlocked ...
unlocked ...
2: received message "hallo" from 1
locked ...
```

Figure 1: A screenshot of the IRV-Tool

at the moment although work is still in progress to extend the functionality of the tool.

4 The IRV-Tool

The IRV-Tool allows to visualize various routing settings that exist in the Internet. With the help of this tool it is possible to setup network topologies with almost arbitrary complexity in a very simple way. The primary goal is to accomplish various simulations of the routing behavior with two different types of routing protocols: RIP, which belongs to distance vector protocol family, and OSPF, which is based on the link state routing algorithm. Hence students can easily learn the basics of dynamic routing with minimal effort.

Our tool was written in Java and works seamlessly in Microsoft Windows as well as the Unix environment. The installation details along with the tool itself can be downloaded from the project website[7]. Once the tool is started, a screen appears as shown in figure 1, except that, in the figure, a topology was already created (or loaded from a file — it is possible to save and load a scenario at any time). The screen consists of a drawing area, where every element is shown together with a number that uniquely identifies it and various other elements that we will briefly outline in what follows.

[7]`http://www.welzl.at/tools/irvtool/`

4.1 User Interface

In terms of the IRV-Tool a network consists of hosts (symbol: computer with screen), routers (symbol: tower) and links (symbol: network cable) which connect two nodes respectively. Here, the term node denotes hosts as well as routers. Hosts can only possess a single connection to a single router and cannot be connected to each other. Routers can possess arbitrary connections to arbitrary nodes. The toolbar on the left hand side (shown in figure 2) serves to design and edit a topology quickly. It consists of certain elements that are used to control the simulation. These are:

Marker tool: used to select or delete a single node or link or to move several nodes together with all the links attached to them.

Remove tool: used to remove several nodes and all the links attached to them by enframing them with pressed left mouse button and releasing the button when ready.

Host tool: used to place a host on the drawing area.

Router tool: used to place a router on the drawing area.

Connection tool: used to connect two nodes to each other.

Edit tool: used to look at and/or modify properties of any entity (node or link). Alternatively, the right mouse button can be used for the said purpose.

The toolbar just below the drawing area controls all facets of simulations within the IRV-Tool. By modifying 'Speed' the execution speed can be adjusted. At 'timeline' the current point in time is recorded (with seconds as unit). By modifying this value and clicking 'apply' the simulation can be moved to the given point in time. Clicking 'Play' starts a simulation, with an animation showing red RIP packets and green standard IP Packets (the various OSPF packet types have colors of their own). 'Pause' pauses a simulation and 'Stop' aborts it while newly initializing all routing-tables (thus it is necessary to reopen a used *.irv file for

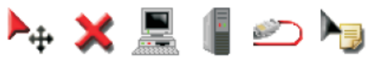
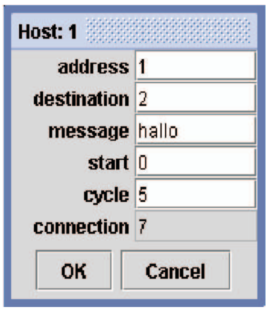


Figure 2: Tools


Host: 1

address	1
destination	2
message	hallo
start	0
cycle	5
connection	7

Figure 3: Host Edit Dialogue

obtaining table entries as stored in the file). The clock (to the far right) can be switched between two modes — either seconds or hh:mm:ss format — by clicking on it.

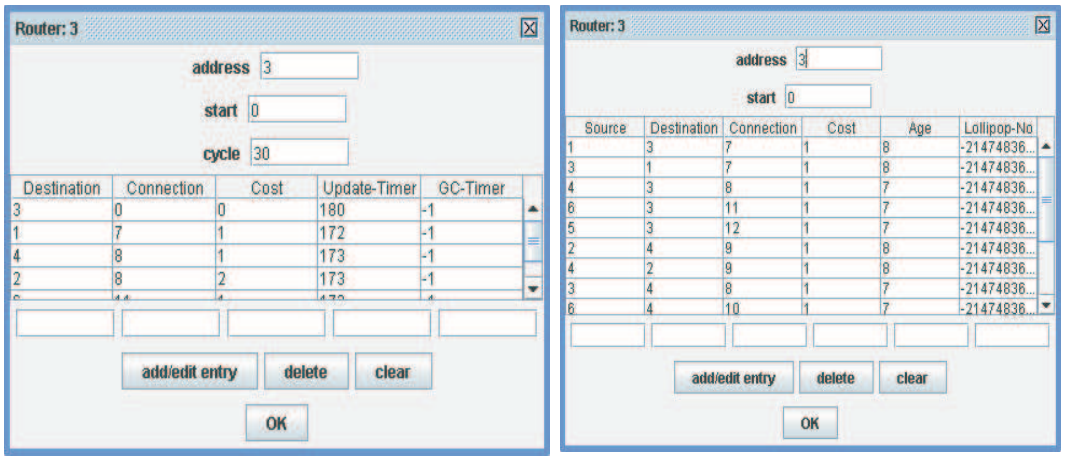
The black area at the bottom of the screen is mainly used for status and error messages. It is also possible to enter commands here, which alleviates the task of creating complex topologies: instead of operating the mouse for every router, a simple loop command can be issued. We do not expect that this functionality will be needed a lot; it is mainly included for teachers who want to create exercises that go beyond the simplicity of the examples that are presented in this paper.

4.2 Edit Options

Hosts: the Host-Edit dialog (figure 3) allows to modify the address (only unambiguous addresses are valid), the destination to which packets are sent, the content of messages from this host, the starting time for sending data packets and the periodicity (every 'cycle' seconds) of these sendings. Moreover, the link to which this host is attached is listed.

Routers: the Router-Edit dialog (figure 4) allows to modify the address, the starting time for updates (-1 means 'never') and their periodicity (only for RIP; default is 30 seconds). Also, the routing table of the selected router can be observed and modified.

Links: the Connection-Edit dialog allows to modify the address and the cost (default value: 1). Via the cost, the distance between two nodes is calculated. If there are two nodes connected by a single link with cost 5, then these nodes are 5 units away from each other (unless there is a shorter path between



address 3

start 0

cycle 30

Destination	Connection	Cost	Update-Timer	GC-Timer
3	0	0	180	-1
1	7	1	172	-1
4	8	1	173	-1
2	8	2	173	-1

address 3

start 0

Source	Destination	Connection	Cost	Age	Lollipop-No
1	3	7	1	8	-21474836...
3	1	7	1	8	-21474836...
4	3	8	1	7	-21474836...
6	3	11	1	7	-21474836...
5	3	12	1	7	-21474836...
2	4	9	1	8	-21474836...
4	2	9	1	8	-21474836...
3	4	8	1	7	-21474836...
6	4	10	1	7	-21474836...

Figure 4: Router Edit Dialogue for distance vector (left) and link state routing (right)

these two nodes). In the drawing area, link costs are displayed in a squared bracket following the unique identifier of the link — in figure 1, only link 11 has costs of 2 and all other links costs are 1. Additionally there is a list of tuples of points in time in the format (MIN, MAX) for each link. Within this list, any number of entries can be stored, indicating from which point in time (MIN) to which other (MAX) this link should be down. Interrupted connections are displayed as broken lines; links that are down behave as if they do not exist.

5 Exercises

Thanks to the save and load capability of the IRV-Tool as it was easy for us to come up with student exercises that are based upon the tool. These exercises are all about distance vector routing — while the link state part of the tool is useful for watching the algorithm in action and obtaining a feel for its dynamic behavior, its implementation still lacks some features that would make exercises interesting enough. This is work in progress. In what follows, we will describe some of our exercises; the “intention” text and text in italics are our comments to the reader and not part of the exercises.

Basic Algorithm Behavior

Intention: to understand how link information is propagated through the network. At the beginning, all the tables are empty; students who solved this exercise will be able to explain why it takes a few iterations for the complete path between the two hosts in the scenario to become available and for the algorithm to adjust to link outage.

Description: Use the IRV-Tool to create the topology in the figure (*the sheet contains a screenshot of the tool which shows a simple topology where three routers are connected to each other, and two of these routers are additionally connected to a host*). Make the following adjustments:

- Let link 10 have a cost of 3, while all other links should have a cost of 1
- Let link 7 go down from time 65 to 125
- Choose the “RIP” routing protocol
- Leave all other parameters as they are

Then answer the following questions:

- What happens during the time period 0 to 65?
- Activate “triggered updates”. What happens now? Why?
- What happens during the time period 65 to 125 (without “triggered updates”)?
- What happens after time 125 (again without “triggered updates”)?

A Strange Problem

Intention: to recognize fundamental limitations of the algorithm. Since “good news” (short distances) overwrites “bad news” in a distance vector protocol, a misconfigured node which informs its neighbors about false “good news” can cause the whole network to fail.

Description: download the file “strangeproblem.irv” from the website. As the name indicates, it illustrates a routing problem. *The file contains a scenario with five routers and two hosts. One of the routers is configured to have costs of 0 to all nodes in the network.* Then answer the following questions:

- What kind of problem is it?
- Can RIP solve the problem? If yes: how, if no: why not?
- Solve the problem by manually altering the routing tables, save the result in a separate file and submit the file.

Bouncing

Intention: to understand the “bouncing” effect and mechanisms that bypass it.

Description: Design a scenario where the “bouncing effect” occurs but “count to infinity” does not. Explain which feature of RIP can be used to overcome the problem and illustrate the process by means of an example. Save your result in a file and submit it.

These relatively easy exercises were suitable in the context of a first basic course on computer networks at the bachelor level — the results of the final exam indicate that distance vector routing was well understood by our students. For higher level courses, we believe that it would be easy to generate much more complex exercises with our tool.

6 Evaluation

From our (teacher’s) perspective, the IRV-Tool worked very well. In order to additionally evaluate the usefulness of our tool from the student perspective, we asked them to fill out an online questionnaire, which was deliberately kept short in order to maximize the number of answers. Since we wanted to make sure that we obtain truthful feedback, the questionnaire homepage contained a link to a free anonymous web direction service[8] for students who might be concerned about their privacy. These are the questions that we asked:

1. Do you consider the tool helpful? Without it, you would have had to solve the problems as theoretical exercises on paper.
2. Now that you solved our exercises, how thorough is your understanding of the contents that we covered with the tool?

[8] `http://www.anonymizer.com`

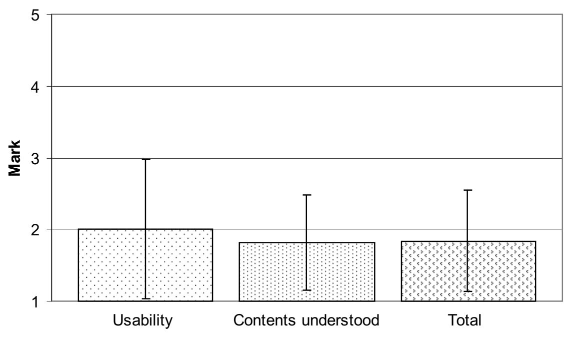


Figure 5: Student feedback

3. How would you rate the usability of the tool?
4. How would you rate the tool on the whole?

The first question was to be answered with “yes” or “no”. Out of the 50 students who filled out our questionnaire, 48 answered it with “yes”. The other questions were to be ranked using our marking system, which ranges from 1 (“very good”) to 5 (“insufficient”). The results are shown in figure 5, where the vertical lines illustrate the standard deviation; given the highly critical attitude that our students are known for (this is indicated by the more general course evaluations that are carried out at our institute every semester), we consider this result highly satisfactory.

In addition to the questions, the questionnaire had an optional field for comments. 13 students used this field:

- One student criticized another part of the course that is not related to the tool.
- Three students commented on minor technical issues that we are now looking at, and did not provide any other feedback.
- The remaining 9 students made very positive comments about the tool. Some examples are:
  - “Using the tool is easy and logical; it could also be used in schools without a problem”

- “It was certainly easier to grasp the contents with the tool than with theoretical exercises on a sheet”
- “All in all a very good program”
- “Nice piece of work, especially split horizon etc. was easier to understand with it”

7 Conclusion

The IRV-Tool has two major capabilities: i) it can animate routing algorithms, and ii) it provides students with the chance to examine their behavior. It is unique in neither of these aspects: algorithm animation has been studied for quite a long time, and not even the ease of creating animations that is attained with the IRV-Tool is too unusual, as there are more generic systems which greatly facilitate the creation of a huge diversity of algorithm animations (cf. [6]). Regarding the chance to study a routing algorithm, there are common simulation packages such as ns-2 or Opnet which offer these possibilities.

What makes the IRV-Tool stand out among these other facilities is its simplicity and its interactive nature. Both aspects are at the core of its design, and students’ feedback clearly indicates that this was a good decision; it clearly showed their disapproval over using ns-2 to teach congestion control mechanism as most of their time and energy was consumed in learning the tool itself. Had we done the same with dynamic routing algorithms, the results would surely have been similar.

While the IRV-Tool has already proven its worth in its current form, we would like to extend its usage to studying problems that can occur with link state routing. Therefore, we are currently extending the OSPF part of the IRV-Tool implementation; at the same time, we are carefully avoiding to add too many bells and whistles such that its main advantage — its ease of use — remains intact. We made the IRV-Tool a permanent part of our teaching plan and hope that, via this paper, we manage to convince some colleagues to do the same.

8 Acknowledgments

The authors would like to thank Christian Sternagel, who implemented the IRV-Tool under our supervision, and the students who answered our questionnaire.

References

[1] Michael Welzl and Max Mühlhäuser, "CAVT: a congestion avoidance visualization tool", *SIGCOMM Comput. Commun. Rev.*, Vol. 33, No. 3, pp. 95-101, 2003.

[2] Michael Welzl, "Network Congestion Control: Managing Internet Traffic", *John Wiley & Sons*, july 2005.

[3] Michael J. Jipping, Andrew Kalafut, Nathan Kooistra and Kathleen Ludewig, "Investigating wired and wireless networks using a java-based programmable sniffer", *ITiCSE '04: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education*, pp. 12-16, NY, 2004.

[4] Wolfgang Schreiner, "A java toolkit for teaching distributed algorithms", *ITiCSE '02: Proceedings of the 7th annual conference on Innovation and technology in computer science education*, pp. 111-115, NY, 2002.

[5] Christian Huitema, "Routing in the Internet", *Prentice Hall*, Second Edition, NewJersey, 2000.

[6] B.A. Colombo, C. Demetrescu, I. Finocchi and L. Laura, "A system for building animated presentations over the web", *Proceedings of the AICCSA '03 Workshop on Practice and Experience with Java Programming in Education*, pp. 31-39, Tunis, july 2003.

Authors: Michael Welzl, Muhammad Ali
(Michael.Welzl,Muhammad.Ali)@uibk.ac.at
Institute of Computer Science, University of Innsbruck,
Techniker Strasse 21-A, 6020, Innsbruck, AUSTRIA.