ARL-SR-0380 • AUG 2017

US Army Research Laboratory

An Extensible NetLogo Model for Visualizing Message Routing Protocols

by Robert P Winkler and Somiya Metu

NOTICES

Disclaimers

ARL-SR-0380 ● AUG 2017

US Army Research Laboratory

An Extensible NetLogo Model for Visualizing Message Routing Protocols

by Robert P Winkler and Somiya Metu
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED (From - To)**
August 2017	Special Report	1–31 December 2016

4. TITLE AND SUBTITLE An Extensible NetLogo Model for Visualizing Message Routing Protocols	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Robert P Winkler and Somiya Metu	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CII-B Aberdeen Proving Ground, MD 21005-5067	**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-SR-0380
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We have developed an extensible NetLogo model for visualizing the operation of different routing protocols in the NetLogo environment. The model can accommodate a wide variety of message routing algorithms by providing methods to answer 2 questions: When 2 couriers come in contact, for each message not already shared, should this particular message be copied and transmitted to the other? If so, how many copies? To demonstrate the generalizability of the model, we present 3 different message-routing protocols: epidemic routing and the “source” and “binary” versions of Spray and Wait.

15. SUBJECT TERMS

message routing, delay tolerant networks, epidemic routing protocol, spray and wait routing protocol

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Robert P Winkler
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	22	301-394-1689

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

This report is intended as supplemental documentation for the unlimited distribution of a NetLogo[1] model for demonstrating and visualizing message routing protocols in intermittently connected networks in the NetLogo[1] environment.

We have developed an extensible NetLogo model for visualizing the operation of different routing protocols in the NetLogo environment. The model can accommodate a wide variety of message routing algorithms by providing methods to answer 2 questions: When 2 couriers come in contact, for each message not already shared, should this particular message be copied and transmitted to the other? If so how many copies? To demonstrate the generalizability of the model, we present 3 different message-routing protocols: epidemic routing[2] and the "source" and "binary" versions of Spray and Wait.[3]

This report is organized as follows. First, we state our contribution. Then we provide a basic description of routing protocols sufficient to distinguish the ones demonstrated here. Next, we present the model interspersed with the relevant NetLogo code and illustrations. This is followed by some basic experiments intended to inform basic intuitions regarding the protocols' operating characteristics rather than support formal analysis. In the Appendix we illustrate and annotate the model's user interface.

2. Contribution

NetLogo is an integrated environment for model development, visualization, and analysis. The ability to visualize models in operation at various speeds and model parameters can be changed during operation to visualize their effects. NetLogo, coupled with the simplicity and ease of use of its agent-oriented programming language, is popular among educators. There are currently hundreds of NetLogo models ranging from fields as diverse as games to the hard sciences to the social sciences to computer-generated art. NetLogo represents the world as a set of "turtles" (nodes in a graph) connected by undirected or directed "links" (edges in a graph) moving about a grid of "patches". The main contribution of this work is to provide an extensible NetLogo model for visualizing a wide variety of routing protocols for instructional purposes. Although we only present topology-free routing algorithms here, the model is easily extended to include topology-informed ones. We leave that as future work.

3. Message-Routing Protocols

The motivation behind message-routing protocols is to efficiently deliver messages from sources to destinations in an intermittently connected network where generally no direct route exists between them. One solution is to consider each encounter as a potential courier if it is not the destination and so on until the message is delivered or expires after a period of time or number of courier hops. Done indiscriminately, this floods the network. While this is the quickest strategy, it also costs the most number of couriers involved. One technique for trading quickly for cost is to introduce a probability controlling the rate of copy transmission. This is called epidemic routing.[2] Another technique is to start with a fixed number of copies and distribute them as couriers are encountered. Once all the copies have been handed out, the model continues to run until all the messages are delivered or expire. This is called spray and wait routing.[3] The distribution of the copies is the "spray" phase followed by the "wait" stage until the message or a copy is delivered. The authors suggest 2 different forms of spray. In "source" or "vanilla" spray and wait (SPWS), copies are distributed one at a time as couriers are encountered until they are exhausted. In "binary" spray and wait (SPWB), half of the copies are distributed to each courier encountered until they are exhausted.

4. The Model

The model consists of messages generated, transported, and received by couriers.

```
breed [messages message]
messages-own [
    src                   ; courier who created the original message
    dst                   ; courier it's going to
    hops                  ; number of hops so far of this copy
    created               ; time of creation of original message
    copy-count]           ; the number copies of this message this courier carrying

breed [couriers courier]
couriers-own [
    cached-messages       ; table of messages indexed by (src,dst) pairs of couriers
    contacts              ; table of contacts for last time contacted (contact only occurs if a message
                          copy transmission occurred)
    last-heading-change]  ; keep track of the last heading change ticks to know when to change direction
```

In the topology-free protocols demonstrated here, each courier has local knowledge of its own position and the messages it carries as well as the messages of any other courier it encounters. It has no access to the position of any other couriers or any other world knowledge. In topology-aware protocols, additional world knowledge is made available locally to inform the distribution of the messages and copies. In our model different routing protocols are incorporated by providing strategies for answering 2 questions: When 2 couriers meet, for each message they do not share, should a copy of the message be given to the other courier? If so how many copies?

A wide variety of routing protocols including both topology-free and topology-aware can be visualized by providing methods to answer those 2 questions.

The basic steps in the model* are as follows:

1) Set up the world

```
to setup-world
   ask patches [
        ifelse pxcor = min-pxcor or pxcor = max-pxcor or pycor = min-pycor or pycor = max-pycor
             ; frame the world with dark red patches
             [set pcolor red - 4]
             ; show the grid
             [sprout-cells 1 [set shape "square 2" set color grey ]]
end
```

2) Set up the couriers and messages. There are 4 types of couriers:

   i. Couriers that are not carrying any messages are represented by white bicycles.

   ii. Couriers that are themselves message destinations are represented by red bicycles.

   iii. Couriers that are carrying messages are represented by white bicycles with a green envelope on top. The relative number of messages the courier is carrying is represented by the hue of green. The darker the hue, the more messages the courier is carrying.

* Only the source code sufficient to describe the model is shown here; for the supporting methods, refer to the source code.

iv. Couriers that are both destinations and are carrying messages for others are represented by a red bicycle with a green envelope on top. The hue of green represents the relative number the courier is carrying.

```
to setup-couriers
  create-ordered-couriers #couriers [
    set cached-messages table:make set contacts table:make set color white set shape "circle 2" set
    size contact-range set label who
    set label-color grey set heading random 360 set last-heading-change 0
    ; make sure we aren't outside of the world
    let xc random-xcor
    while [(xc < (min-pxcor + size)) or (xc > (max-pxcor - size))] [set xc random-xcor]
    let yc random-ycor
    while [(yc < (min-pycor + size)) or (yc > (max-pycor - size))] [set yc random-ycor]
    ; set courier initial position
    setxy xc yc]
end
```

Figure 1 shows the world state after a random setup with 3 messages and 9 couriers.

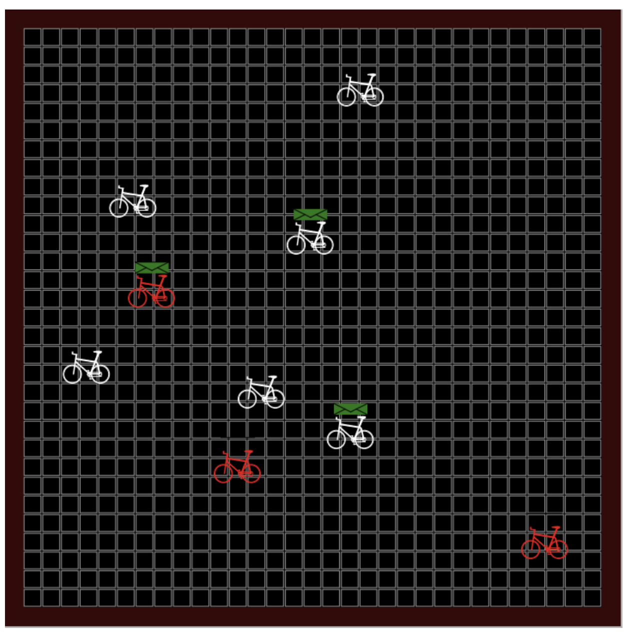


Fig. 1 Grid displaying 3 messages and 9 couriers, one of which is itself a destination carrying messages for other couriers

3) Move about.*

```
to move
   if ticks - last-heading-change > #steps [set heading random 360 set last-heading-change ticks]
   bounce
   let xc xcor + dx
   let yc ycor + dy
   ifelse (xc < min-pxcor)[set xc min-pxcor][if xc > max-pxcor [set xc max-pxcor]]
   ifelse (yc < min-pycor)[set yc min-pycor][if yc > max-pycor [set yc max-pycor]]
   setxy xc yc
   let my-contacts contacts
   ask other couriers [
       ; anyone close?
       if (distance myself < size) [
           ifelse not table:has-key? contacts [who] of myself
           [contact myself]
           [if table:get contacts [who] of myself - table:get my-contacts who > wait-time-between-
           contact
           [contact myself]]]]
   set #steps-this-run #steps-this-run + 1
end
```

4) Contact made.

```
to contact [other-courier]
  ; check the other-courier messages first.
  process-inbox other-courier
  ; now check my messages against the other guy's.
  process-outbox other-courier
end
```

i. Check if you have messages I do not.

```
to process-inbox [other-courier]
       let me self
       let other-table [cached-messages] of other-courier
       let my-table cached-messages
       foreach table:keys other-table [ [x] ->
           let other-msg table:get other-table x
           ; do i already have the message?
           if not table:has-key? my-table x  [
           ; i don't have the message. is the new message for me?
           ifelse [dst] of other-msg = [who] of me
                   ; i've got mail
                   [ask other-courier [ deliver other-msg ]]
                   ; new message is not for me. Shall i courier it?
                   [if [xmt?] Of other-msg [ask other-courier [xmit me other-msg]]]]]
end
```

ii. Now check whether I have messages you do not.

```
to process-outbox [other-courier]
       let me self
       let courier-table [cached-messages] of other-courier
       foreach table:keys cached-messages [ [x] ->
           let my-msg table:get cached-messages
           ifelse [hops] of my-msg > max-hops – 1
                   [ask my-msg [drop]]
                   [ifelse ticks - [created] of my-msg > message-timeout
                       [ask my-msg [timeout]]
                       ; Does the other guy already have this message?
```

* Change this method to add different movement models.

```
[if not table:has-key? courier-table x [
      ; Nope. Is the message for the other guy?
      ifelse [dst] of my-msg = [who] of other-courier
      ; My message is for the other courier, deliver it.
      [deliver my-msg]
      ; My message isn't for the other guy. Will they courier it?
      [if [xmt?] of my-msg [xmit other-courier my-msg]]]]]]
end
```

iii. Deliver the messages if appropriate.

```
to deliver [msg]
      set #delivered #delivered + 1
      set delay delay + ticks - [created] of msg
      ; Update contacts.
      table:put contacts [who] of courier [dst] of msg ticks
      ask courier [dst] of msg [table:put contacts [who] of myself ticks]
      ; Paint him done.
      ask courier [dst] of msg [set-shape-color]
      ask msg [GC]
      ask couriers [set-shape-color]
end
```

iv. Courier the message as appropriate.

a. Can the message be transmitted?*

```
to-report xmt?
      let ret false
      let num random-float 1.0
      if table:length [cached-messages] of courier dst > buffer-size - num-to-copy
      false [report false] ; buffer full
      ifelse strategy = "epidemic" and num <= xmit-probability
      [set ret true]
      ; for SPW, when the message's copy gets to 1, don't transmit it further
      [if copy-count > 1 [set ret true]]
      report ret
end
```

b. Exchange messages

```
to xmit [to-courier msg]
      ; Ask the receiver to make a copy of the message.
      ask msg [set hops hops + 1]
      ask to-courier [
            let copy clone (msg)
            ask copy [set copy-count num-to-copy false] ; receiver
            table:put cached-messages [key] of copy copy
            ; paint him
            set-shape-color]
      ; Update  copies.
      ask msg [set copy-count num-to-copy true] ; i am sender
      ; Paint me.
      set-shape-color
      if shape != "target" [set shape "circle"]
      set color scale-color green [copy-count] of msg (#copies * 1.25) 0
      ; Update stats.
      set #xmit #xmit + 1
      ; Update contacts.
      table:put contacts [who] of to-courier  ticks
      ask to-courier [table:put contacts [who] of myself ticks]
end
```

* Modify this method to add new routing protocols.

Figures 2 and 3 show the network state after some exchanges have occurred. The hue of green represents the relative number of message copies each courier is carrying. The darker the hue, the more copies.

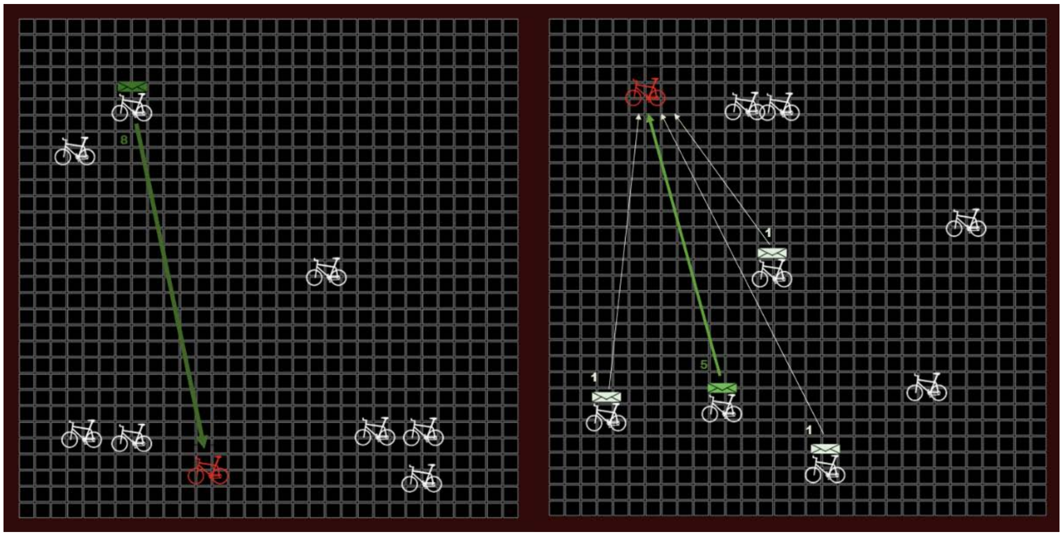


Fig. 2 SPWS with 9 couriers and 1 message before and after some exchanges. Since epidemic routing and SPWS both distribute 1 copy at a time, they are visually indistinguishable until the number of epidemic routing copies exceeds SPWS.

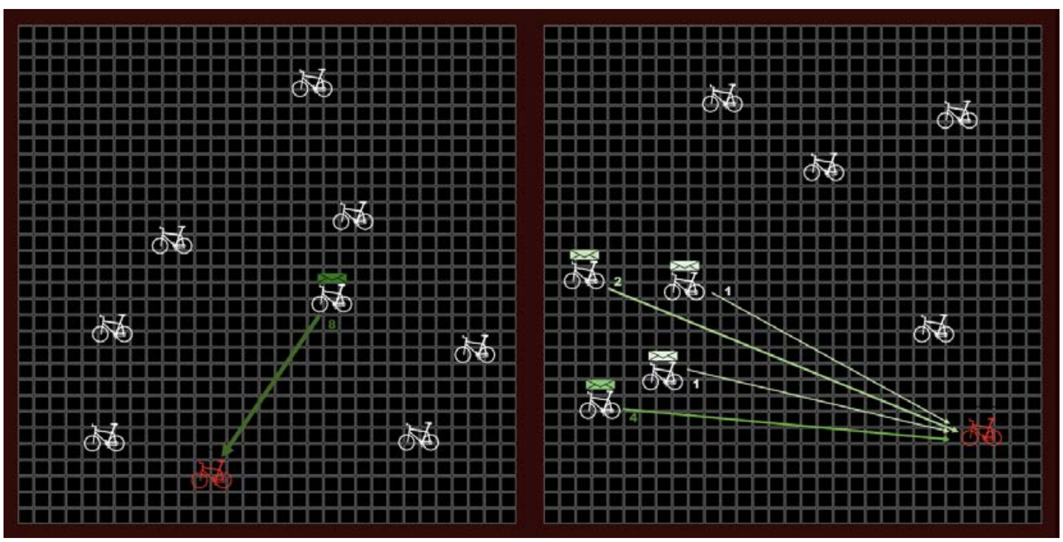


Fig. 3 SPWB with 9 couriers and 1 message before and after some exchanges

5) Repeat until there are no longer undelivered messages.

```
to go
    ask couriers [move]
    if (count messages = 0) [
        setup-run
        set update? True
```

Approved for public release; distribution is unlimited.

```
            ask couriers [die]
            ask messages [die]
            reset-ticks
            wait 0.25
            setup-couriers
            setup-messages]
        tick
        if ticks - last-inject > time-to-inject [inject]
        if run# > #runs [stop]
    end
```

Next, we present some basic statistics collected for the different protocols. As mentioned previously, the intent of these statistics is not for detailed formal analysis but rather to inform basic intuitions of the routing protocol operating characteristics.

5. Experimental Results

The data were collected on a 200 × 200 grid with 400 couriers randomly situated. The couriers' contact range was set to one grid cell. One message was randomly injected per run. The buffer size was set so that any courier could handle the maximum number of copies. Message timeout and the maximum number of hops before dropping were set high enough to ensure neither would occur. The movement model was random walk with one step between random direction changes. One hundred simulations were run for each configuration. The data collected were mean message delay and mean message transmissions per run. The number of message copies for SPWS and SPWB was varied from 8, 16, 32, 64, 128, and 256. The forwarding probability for epidemic routing was set to the ratio of the number of SPW copies to the number of couriers. Table 1 and Fig. 4 show the average delay for the routing protocols based on the number of copies and forwarding probability as specified previously. Table 2 and Fig. 5 show the average transmission cost by routing protocol.

Table 1 Average delay

Copies	p(xmit)	SPWB	SPWS	Epidemic
8	0.02	3052	3392	4090
16	0.04	3049	3549	3028
32	0.08	2905	3127	2196
64	0.16	2417	3911	1648
128	0.32	2323	3632	1294
256	0.64	1938	3291	1212

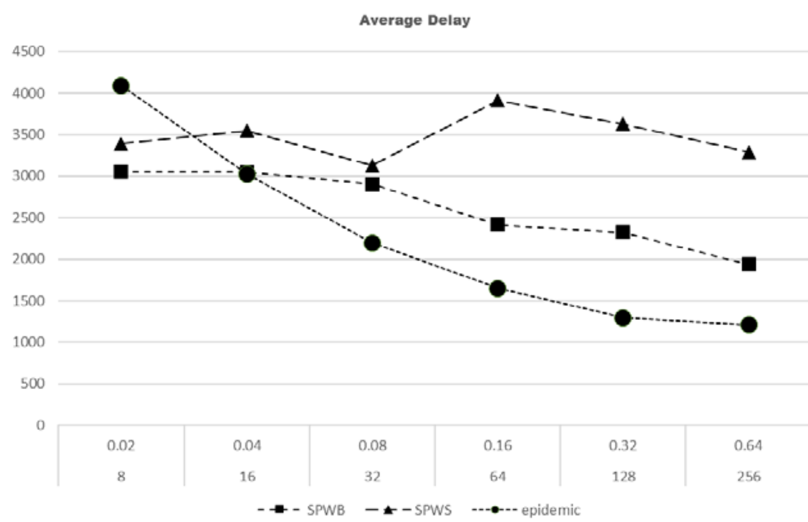


Fig. 4 Average delay

Table 2 Average number of transmissions

Copies	p(xmit)	SPWB	SPWS	Epidemic
8	0.02	6.9	7	70.3
16	0.04	14.2	14	129.2
32	0.08	29.4	25	161.1
64	0.16	49.9	39.6	173
128	0.32	89.2	38.7	185.7
256	0.64	113.7	37.8	205.5

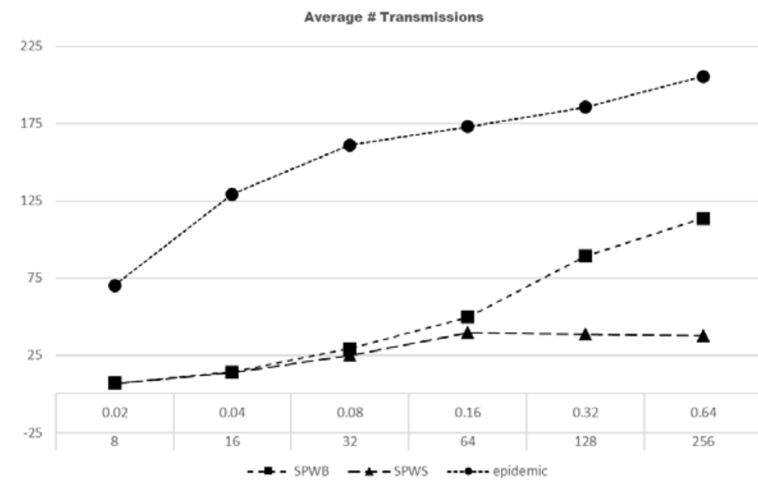


Fig. 5 Average number of transmissions

Starting at 8% transmission probability, epidemic routing consistently delivers messages faster than both SPWS and SPWB by anywhere from 30%–65% for SPWS to 25%–45% for SPWB. But it does so at an increased transmission cost of anywhere from 4–10 times SPWS and 2–10 times SPWB. This increase in cost is expected given that epidemic routing is unconstrained in the number of copies it can make.

Intuition 1: If cost is not a factor, prefer epidemic routing.

There is little difference between SPWB and SPWS in terms of delay until 64 copies or 16% of the courier population. After that, SPWB outperforms SPWS by approximately 40% at an increased transmission cost of 1–3 times.*

* SPWB could have as many as half the number of copies of couriers distributing messages while SPWS always has only one.

Intuition 2: If cost is a factor but not the primary one, prefer SPWB over epidemic routing and SPWS.

SPWS always costs the same or less than SPWB, and after the number of copies exceeds approximately 15% of the courier population, the transmission cost of SPWS is invariant.

Intuition 3: If cost is the primary consideration, prefer SPWS over epidemic routing and SPWB.

6. Conclusion

In this report we have presented an extensible NetLogo model for demonstrating and visualizing a variety of different classes of message-routing protocols. New protocols can be added by providing methods that answer 2 questions: When 2 couriers come in contact, for each message not already shared, should this particular message be copied and transmitted one to the other? If so how many copies? Additional movement models can be accommodated by overriding the courier "to move" command. We demonstrated how the model can be used to inform basic intuitions regarding the message-routing protocol operating characteristics.

The purpose of this report is to provide supplemental information in support of distribution of the source code for the model back to the NetLogo community. The main contribution is an extensible model for demonstrating, evaluating, and visualizing a variety of classes of routing protocols in the NetLogo environment.

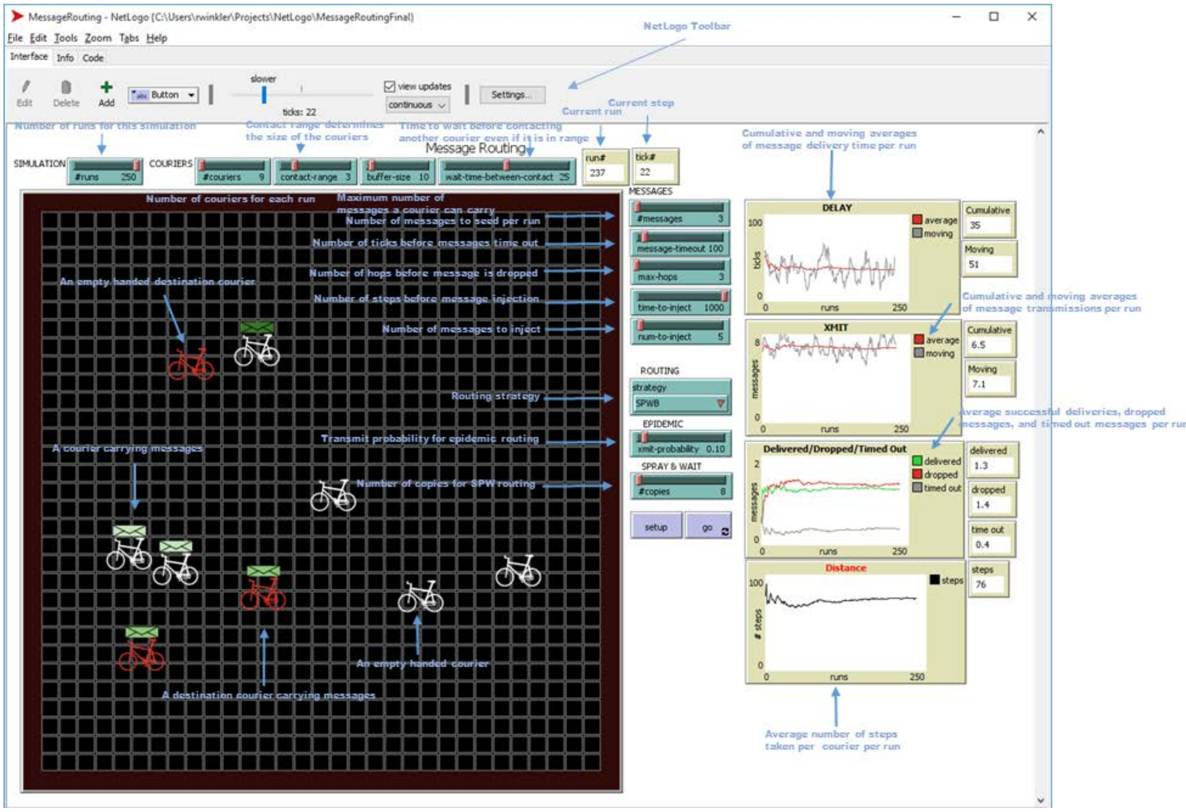
7. References

1. Wilensky U. NetLogo. Evanston (IL): Center for Connected Learning and Computer-Based Modeling, Northwestern University; 1999 [accessed 2017 Feb 15]. http://ccl.northwestern.edu/netlogo/.

2. Vahdat A, Becker D. Epidemic routing for partially connected ad hoc networks. Durham (NC): Duke University; 2000. Report No.: CS-200006.

3. Spyropoulos T, Psounis K, Raghavendra CS. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. Proceedings of the 2005 ACM SIGCOMM workshop on delay-tolerant networking (WDTN '05); 2005 Aug 22–26; Philadelphia, PA. p. 252–259.

Appendix. Message Routing Protocol NetLogo Model User Interface



Approved for public release; distribution is unlimited.

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	DIR USARL RDRL CII B R WINKLER

INTENTIONALLY LEFT BLANK.