

Emulating sequential scanning worms on the DETER testbed

George Kesidis, Lunquan Li, Ihab Hamadeh, Soranun Jivasurat,
Peng Liu
Penn State University

Open-source code downloadable at
[Http://emist.ist.psu.edu](http://emist.ist.psu.edu)

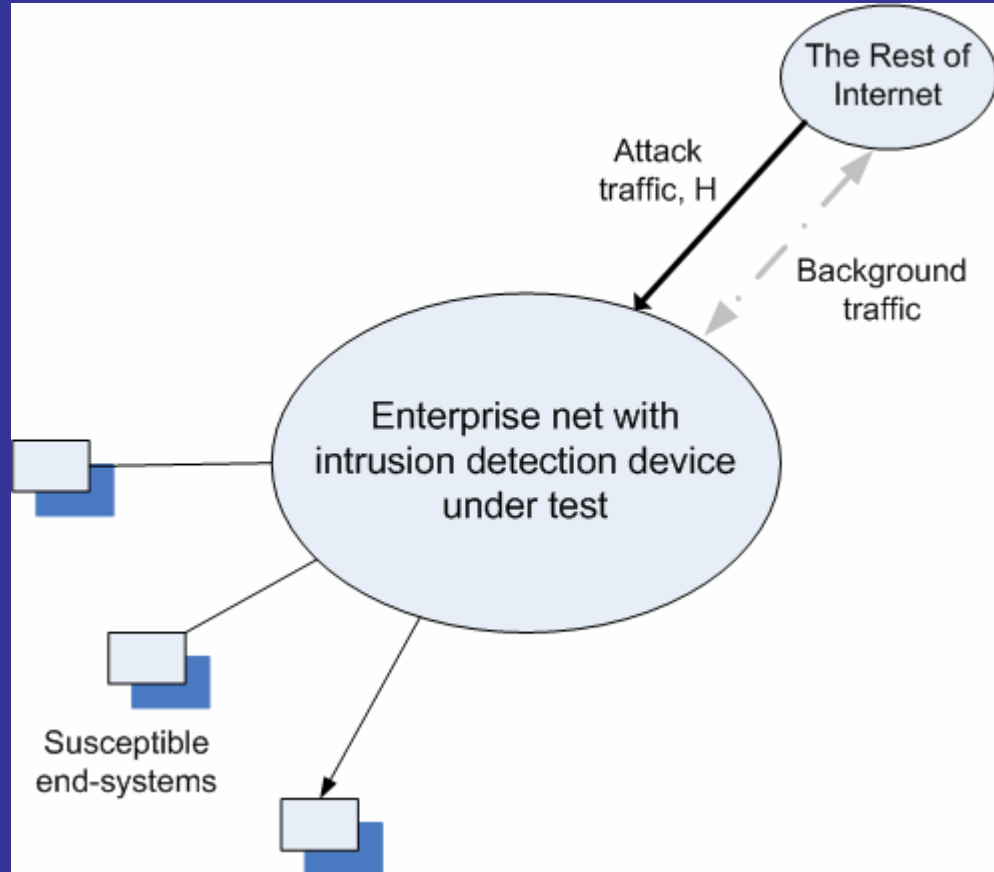
DETER Community Workshop
Arlington, VA, June 15,16, 2006

1. Experimental methodologies for Worm research

- Mathematical analysis and modeling
- Simulation: By a single CPU or a distributed simulator
- Emulation: 1:1 on a test-bed
- Hybrid: Combination of simulation and emulation

Experiment Overview

- Replay the worm propagation in high fidelity using test bed (DETER)
- Understand the impact of worms on enterprise networks
- Build foundations for worm defense experiments



1K node experiment

2. Worm attack traffic modeling: KMSim-II

- **KMSim model: a variation of kermack-Mckendrick model**

- J be the number of different groups of peripheral enterprise networks in the Internet,
- σ_{ji} represent the total scanning rate *out* of the enterprise to the rest of the Internet of a group- j enterprise network with i infectives (i.e., infection-level i)
- $y_{j,i}(t)$ be the number of group- j enterprises with infection-level i ,
- $C(j)$ be the maximum infection level of group- j enterprises, and
- the total scan rate *to the Internet* of the worm be

$$S(t) \equiv \sum_{j=1}^J \sum_{i=1}^{C(j)} \sigma_{j,i} y_{j,i}(t).$$

$$\dot{y}_{j,C(j)}(t) = \beta_{j,C(j)-1} y_{j,C(j)-1}(t), \quad (1)$$

$$\dot{y}_{j,i}(t) = (\beta_{j,i-1} y_{j,i-1}(t) - \beta_{j,i} y_{j,i}(t)) \quad i \in [1, C_j) \quad (2)$$

$$\dot{y}_{(j,0)}(t) = -\beta_{j,0} y_{j,0}(t). \quad (3)$$

KMSim-II adds worm death/removal:

$$dy_{j,i}/dt \quad - = \quad \delta_i y_{j,i}$$

$$dy_{j,i-1}/dt \quad + = \quad \delta_i y_{j,i}$$

where the removal/death rate is $\delta > 0$ and $\delta_i \equiv i\delta$.

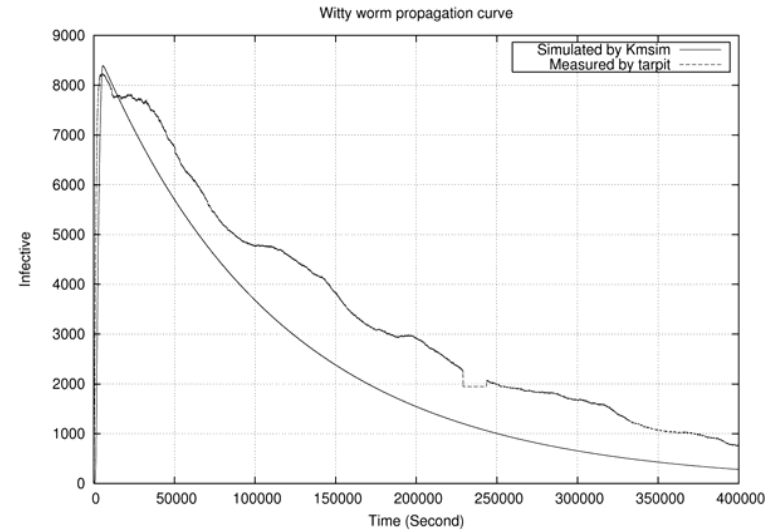
2. 1 KMSim-II: Witty and Blaster simulation

Algorithm 1 Removal/Death of the Infected and Susceptible

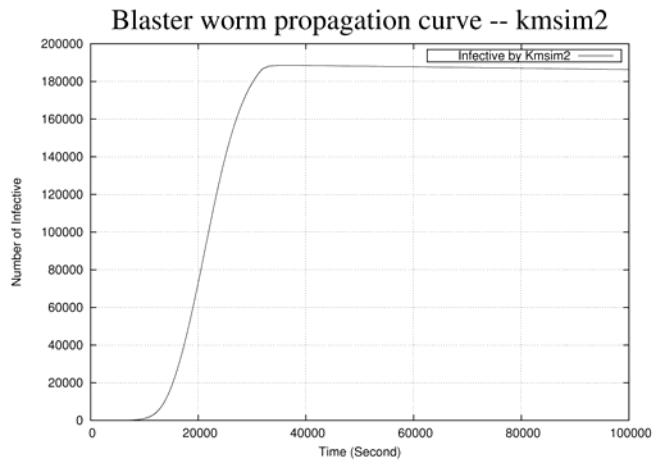
```

1: for j do
2:    $newdeath \leftarrow \delta * ce_j * y_{j,ce_j} * dt$ 
3:    $death_j += newdeath$ 
4:    $y_{j,ce_j} -= newdeath$ 
5:   for i =  $ce_j - 1$  downto 1 do
6:      $y_{j,i} += newdeath$ 
7:      $newdeath \leftarrow \delta * i * (y_{j,i} - newdeath) * dt$ 
8:      $y_{j,i} -= newdeath$ 
9:   end for
10: end for

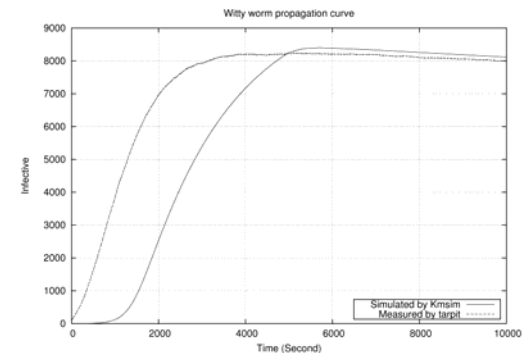
```



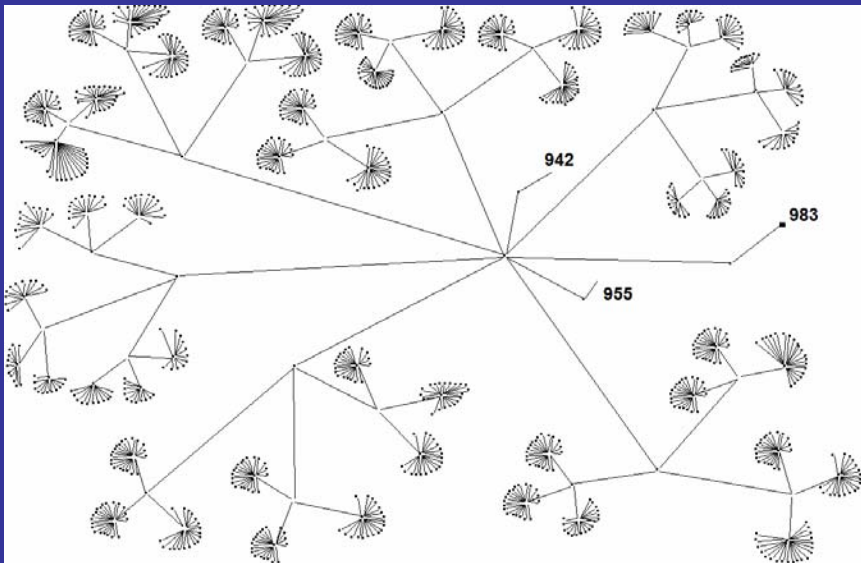
Parameters: vulnerable population 12,000
 $j=1$, $c=4$, $NE=3,000$
 $\sigma = [1800, 2400]$



Blaster similarly simulated and used in the following emulation



3. Emulation challenge : scaling down



1002 nodes

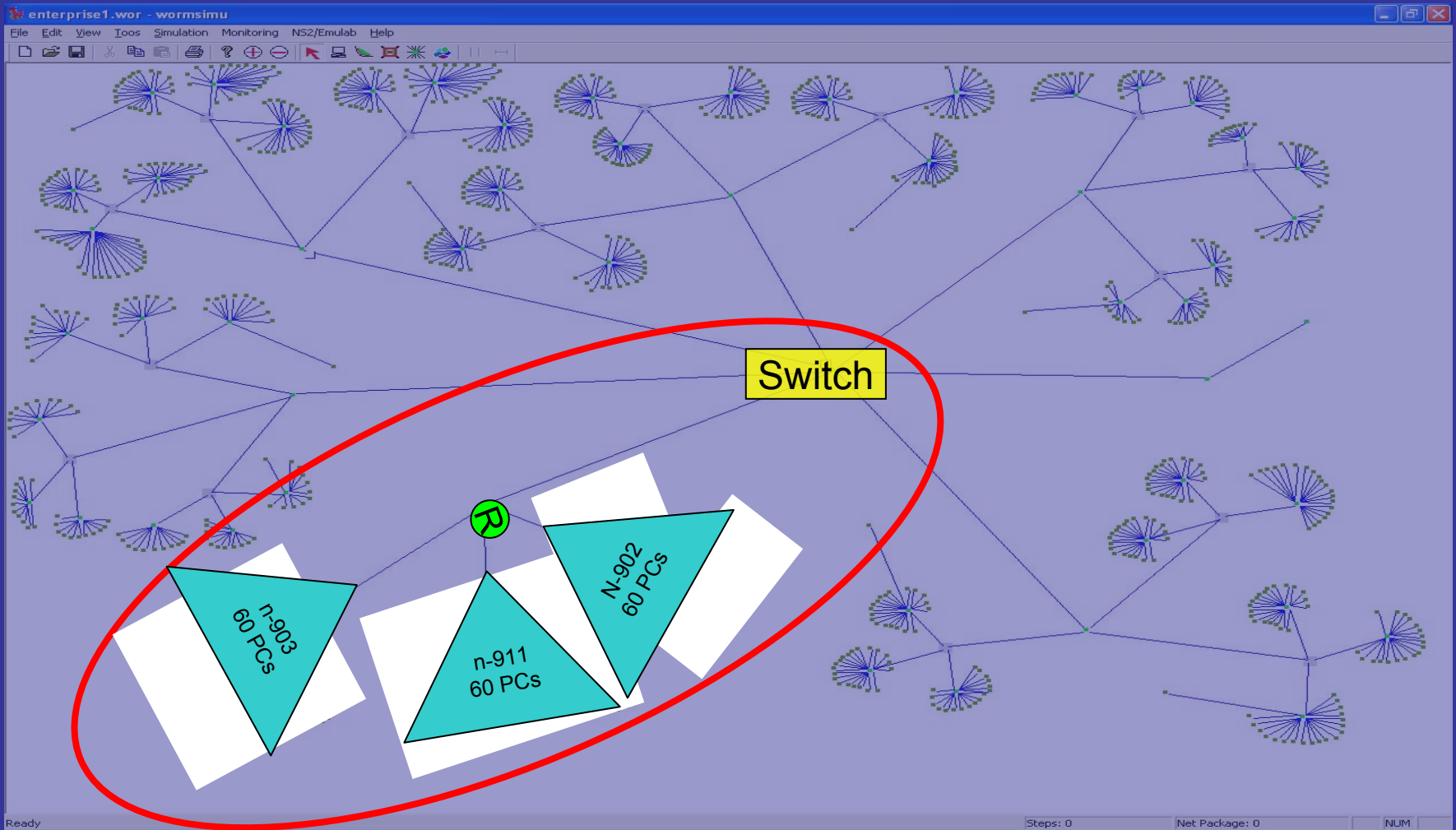


DETER: 2xx nodes

So virtualization-based abstraction is necessary!

Requirements: (1) scalability
(2) graceful degradation of fidelity

3. 1 Solution: Virtual node approach



3.2 Virtualization Comparison

	VMware	Emulab VM	NS2	Virtual Node
OS Platform	Windows & Linux	FreeBSD	Linux/FreeBSD	Linux/FreeBSD
Test-bed Support	No	Yes	Yes (NSE)	Yes
Scale of Virtualization	4-8	<20	100-1000	20-200
Supporting unmodified Apps	Yes	Yes	No	No
Fidelity	High	?	?	?

DETER Experiment using ESVT

Step 1. Setup the experiment using the EMIST GUI

- EMIST topology specification in TCL
 - Virtual sub-network nodes
 - Internet interface
 - Normal & vulnerable nodes
 - Bandwidth, latency, addresses, OS
- Other auxiliary TCL scripts

Step 2. Setup the DETER environment

- Worm program
- Traffic generator program
- Internet interface program
- Virtual node program
- Normal node program
- Vulnerable node program
- TCPDUMP setup
- EMULAB GUI can be used here

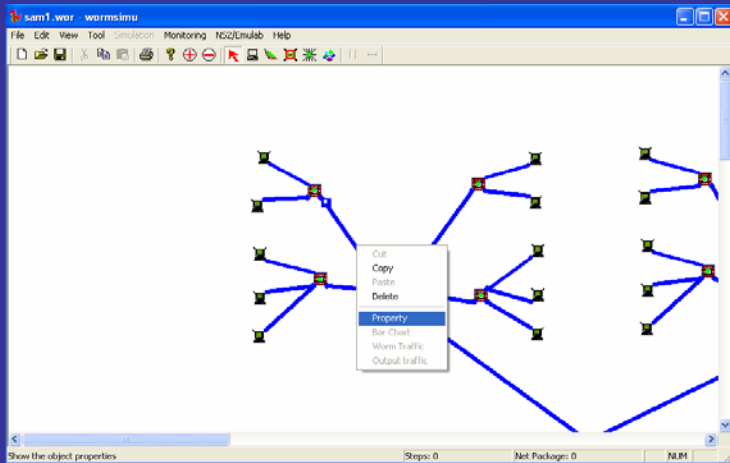
Step 3. Run the experiment on DETER

Step 4. Visualize the results using the EMIST GUI

- Worm propagation snapshots
- Worm propagation animation
- Link traffic bar chart (dynamic)
- Worm replay

4. ESVT (Experiment Specification and Visualization Toolkit)

Topology editing



TCL Script Generation

Global component and script property configuration

Components Property

Make change on: Selected ALL

Use FREEBSD by default

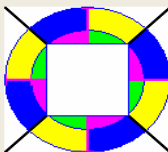
Computers/Hosts % Vulnerable

Link Bandwidth kbps

Experiment Duration: Seconds

Node Startup Command Directory:

Switch Property



Index:

This version of switch has a lot bandwidth, it will not cost any delay. If you want a delay or other property, change the corresponding link properties.

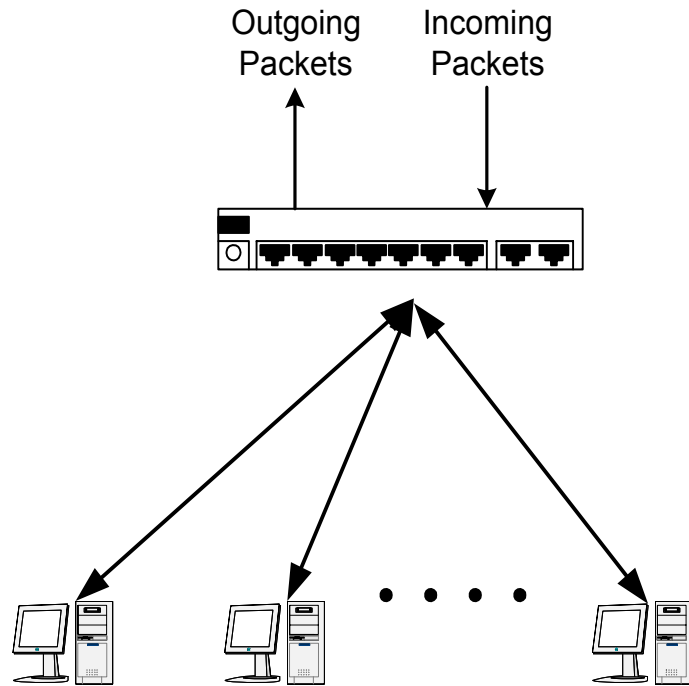
Switch Links

COMPUTER:159	COMPUTER:181
COMPUTER:164	COMPUTER:182
COMPUTER:165	SWITCH:15
COMPUTER:170	
COMPUTER:171	
COMPUTER:174	
COMPUTER:176	
COMPUTER:179	
COMPUTER:180	

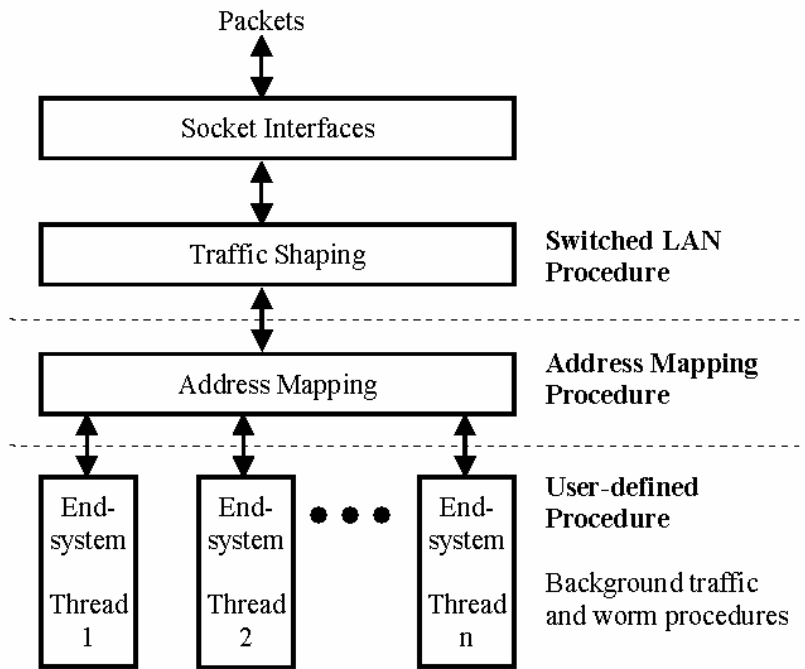
Simulated Lan?

```
##--Total Switch: 3, Computer: 60, Susceptible ones: 0.
set lan70 [$ns make-lan "$n(969) $n(978) " 100Mb Oms]
##--Total Switch: 3, Computer: 58, Susceptible ones: 1.
set link969 [$ns duplex-link $n(979) $n(977) 100Mb Oms DropTail]
# Running programs section
tb-set-node-startcmd $n(902) "/proj/worm/e1k/run_virtual n-902-lan3"
tb-set-node-startcmd $n(903) "/proj/worm/e1k/run_virtual n-903-lan4"
tb-set-node-startcmd $n(911) "/proj/worm/e1k/run_virtual n-911-lan12"
tb-set-node-startcmd $n(913) "/proj/worm/e1k/run_virtual n-913-lan14"
tb-set-node-startcmd $n(916) "/proj/worm/e1k/run_virtual n-916-lan17"
tb-set-node-startcmd $n(921) "/proj/worm/e1k/run_virtual n-921-lan22"
tb-set-node-startcmd $n(924) "/proj/worm/e1k/run_virtual n-924-lan25"
tb-set-node-startcmd $n(927) "/proj/worm/e1k/run_virtual n-927-lan28"
tb-set-node-startcmd $n(932) "/proj/worm/e1k/run_virtual n-932-lan33"
tb-set-node-startcmd $n(936) "/proj/worm/e1k/run_virtual n-936-lan37"
tb-set-node-startcmd $n(943) "/proj/worm/e1k/run_virtual n-943-lan44"
tb-set-node-startcmd $n(945) "/proj/worm/e1k/run_tcp 945"
tb-set-node-startcmd $n(946) "/proj/worm/e1k/run_virtual n-946-lan47"
tb-set-node-startcmd $n(947) "/proj/worm/e1k/run_virtual n-947-lan48"
tb-set-node-startcmd $n(951) "/proj/worm/e1k/run_virtual n-951-lan52"
tb-set-node-startcmd $n(956) "/proj/worm/e1k/run_virtual n-956-lan57"
tb-set-node-startcmd $n(962) "/proj/worm/e1k/run_virtual n-962-lan63"
tb-set-node-startcmd $n(966) "/proj/worm/e1k/run_virtual n-966-lan67"
tb-set-node-startcmd $n(969) "/proj/worm/e1k/run_virtual n-969-lan70"
tb-set-node-startcmd $n(972) "/proj/worm/e1k/run_tcp 972"
tb-set-node-startcmd $n(973) "/proj/worm/e1k/run_tcp 973"
tb-set-node-startcmd $n(974) "/proj/worm/e1k/run_tcp 974"
tb-set-node-startcmd $n(975) "/proj/worm/e1k/run_tcp 975"
tb-set-node-startcmd $n(976) "/proj/worm/e1k/run_tcp 976"
tb-set-node-startcmd $n(977) "/proj/worm/e1k/run_tcp 977"
tb-set-node-startcmd $n(978) "/proj/worm/e1k/run_tcp 978"
tb-set-node-startcmd $n(979) "/proj/worm/e1k/run_internet 979"
$ns rtproto Static
$ns run
```

5. Virtual node Design



Physical topology



Programming Implementation

5.1 Address mapping and packet structure

The virtual header is located inside the payload of IP packets

IP Header	SourceAddr	SourcePort	Pad
	DestinationAddr	DestinationPort	
	Length	Type	

Virtual dest addresses are looked up in the mapping table to locate the address and port number of each virtual host, E.G.,

Network Address	DETER Node Address
10.1.3.0/24	10.1.3.2
10.1.4.0/24	10.1.4.2
10.1.5.0/24	10.1.5.2
10.1.6.0/24	10.1.6.2
default	10.1.8.2

5.2 LAN traffic shaping

- Switched LAN implemented, CSMA/CD LAN and WLAN in the future
- Token-bucket traffic regulator
- Let C be the uplink bandwidth, n be the number of hosts, R be the full transmission speed of any host, the LAN throughput is

$$S(n) = \min \{ R \cdot n, C \},$$

The maximum sending speed per host is $s(n)/n$.

5. 3 Blaster Emulation: packet crafting

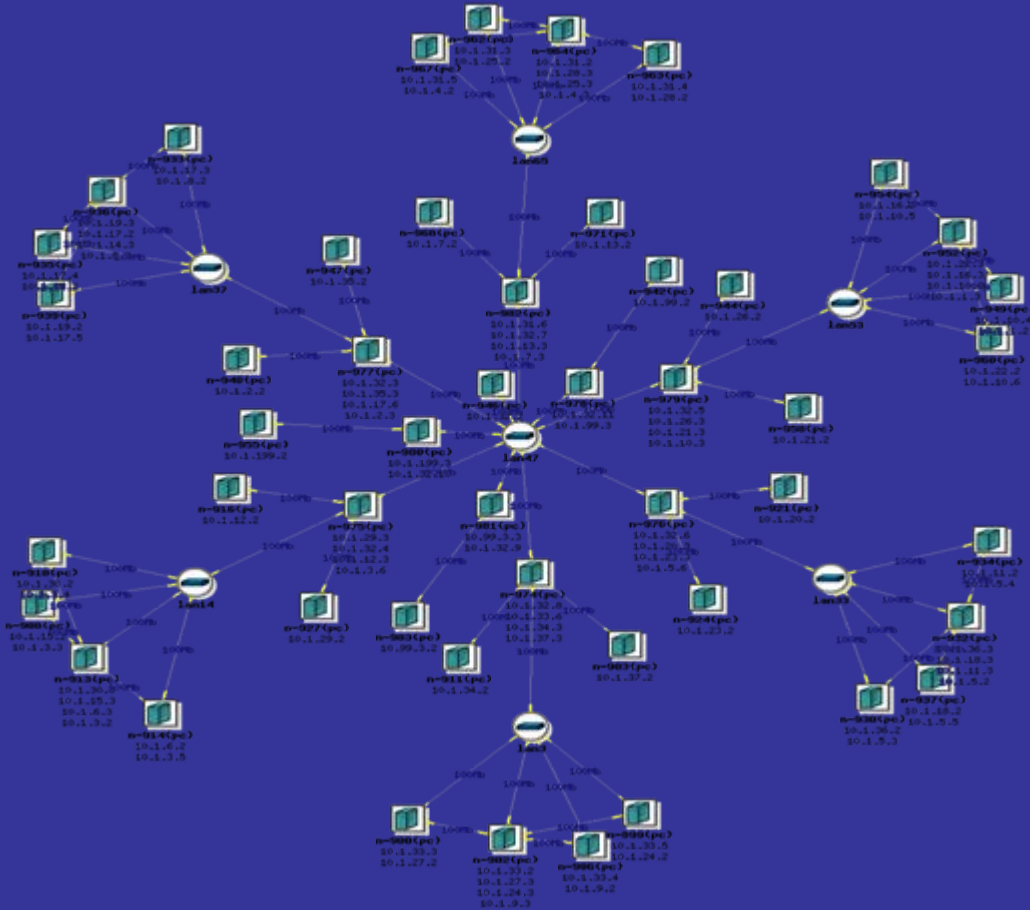
- Using UDP to simulate TCP worm
- Internet injection scan rate is based on KMSim simulation results
- “Sequential” scanning behavior is reserved in Internet scan injection

Algorithm 2 Sequential Scanning Worm Body

```
repeat
2:   for  $j = 1$  to 20 do
      Calculate Next IP  $tip_j$ 
4:    $pkt.destinationIP = tip_j$ 
      Send UDP packet
6:   end for
      Sleep 1.8 seconds
8:   for  $j = 1$  to 20 do
      if  $tip_j$  is infected then
10:     $pkt.payloadlength = BLASTERSIZE$ 
         $pkt.destinationIP = tip_j$ 
12:    for  $l = 0$  to 8 do
          Send UDP packet {Simulate the actual worm infection}
14:     $SendingDelay = (wormpacketsize + 28) * 80;$ 
          Delay  $SendingDelay$ 
16:    end for
          Sleep 1 second
18:    end if
      end for
20: until Program Is Active
```

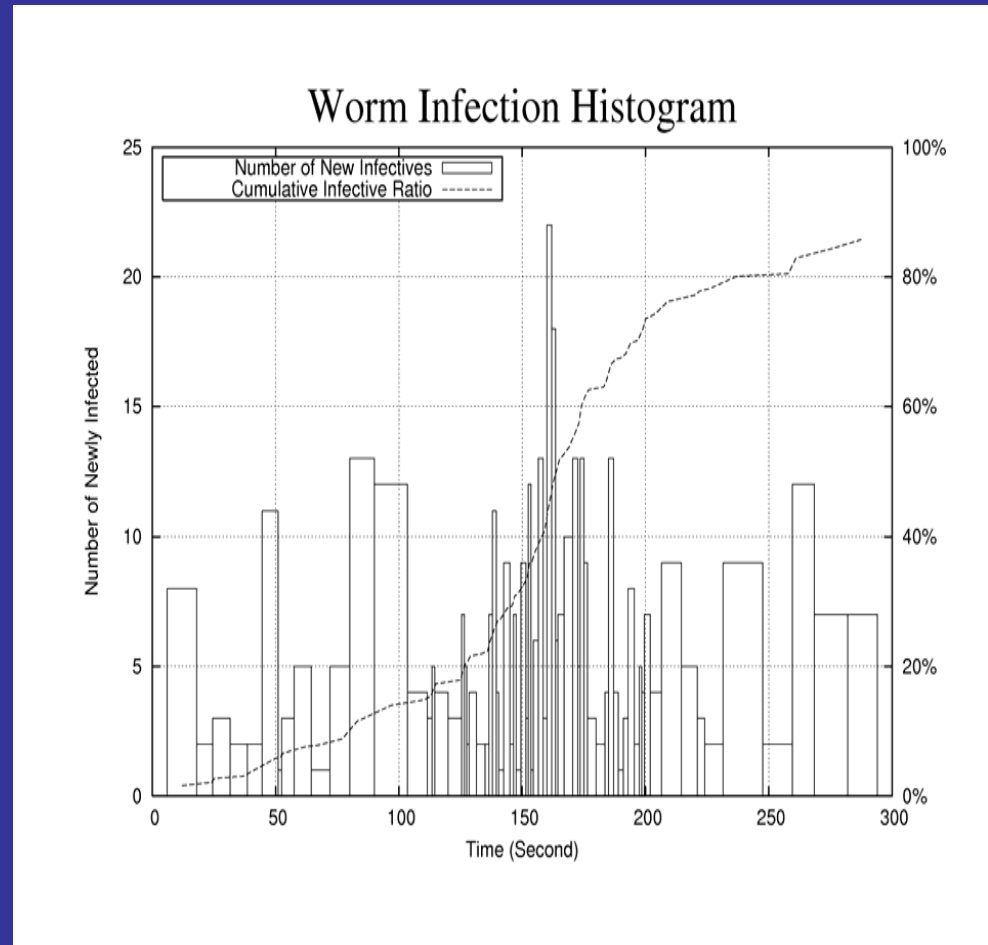
6. Experiment on DETER

- 1000-plus hosts
- experiment takes 49 test-bed nodes
- Emulation lasts for 600 seconds



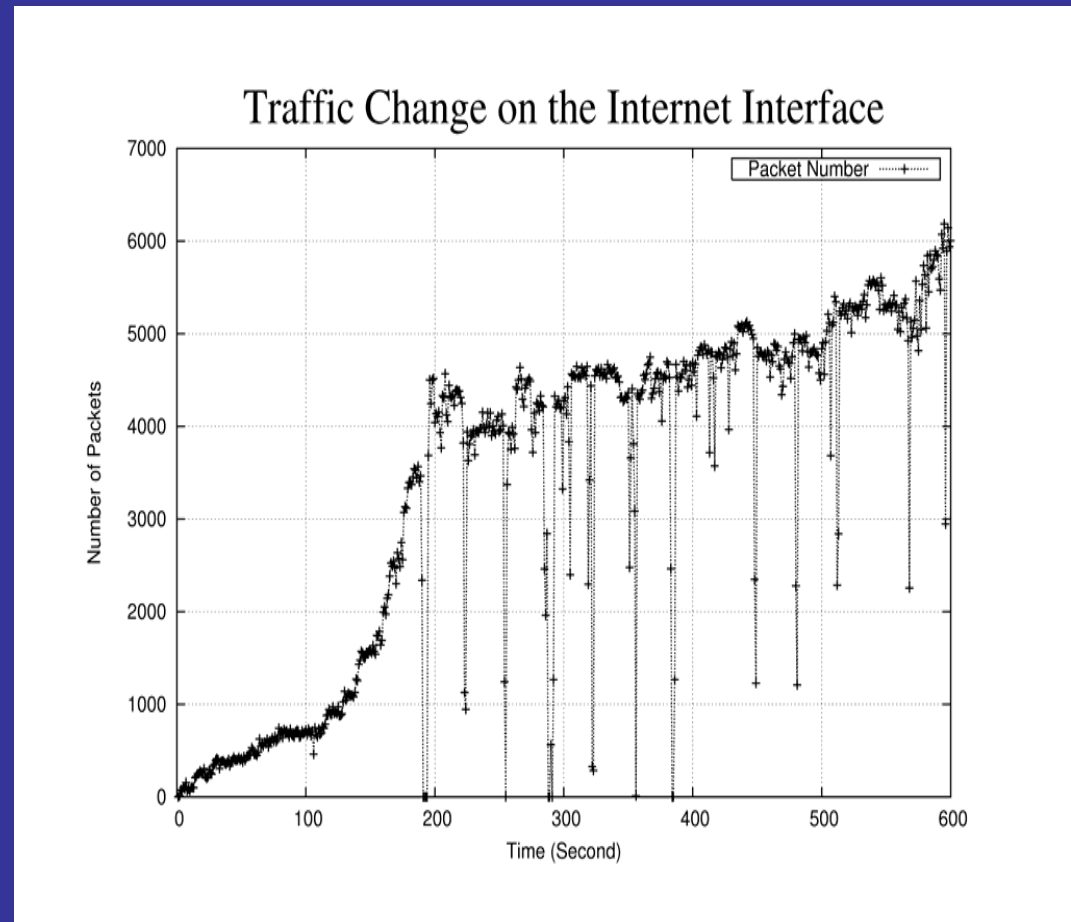
6.1 Blaster Emulation Results

- Emulation: 600 Seconds; 1,400 files and 1GB trace logs
- Infection: 100-400 infected out of 400 plus susceptible nodes

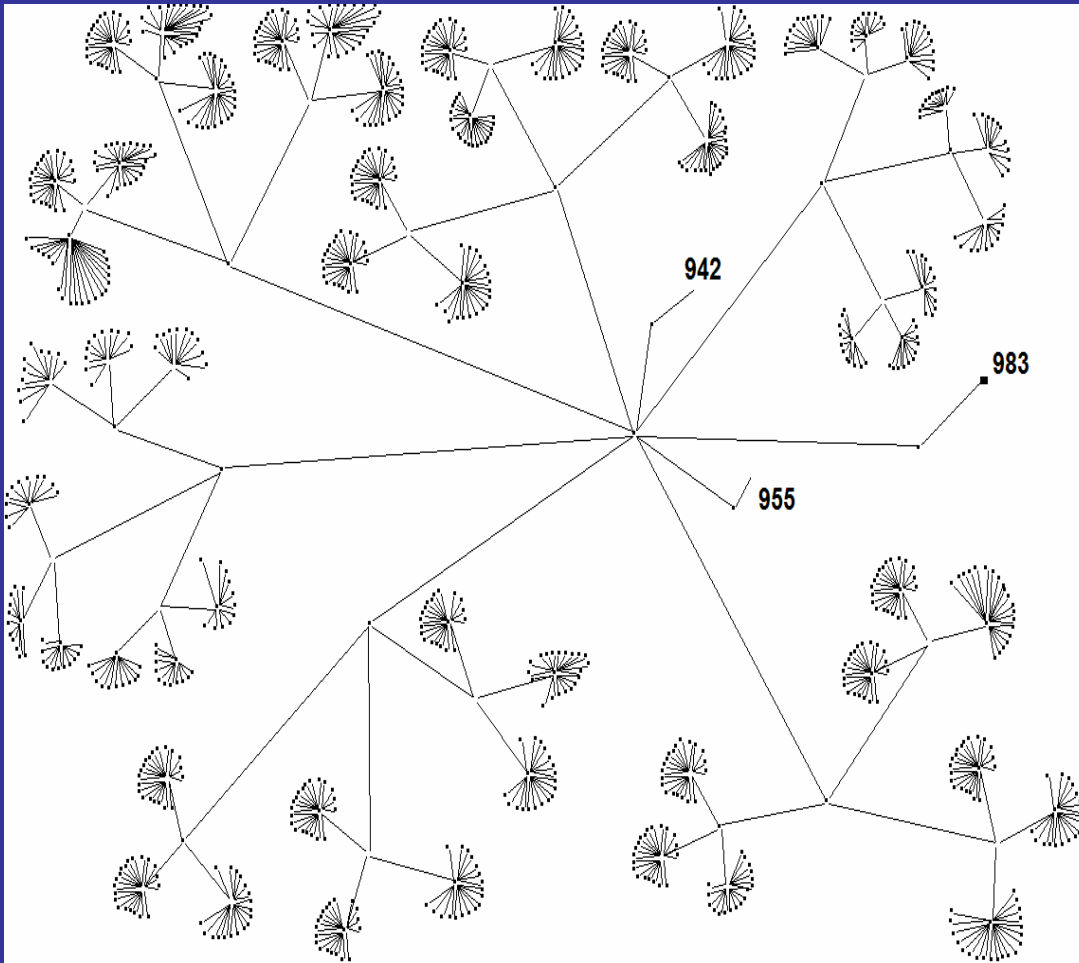


6.1 Blaster Emulation Results

- Characteristics of Blaster: fast local propagation (83% infected between 120sec and 180sec)
- Egress traffic roughly proportional to the number of infected hosts (before saturation)



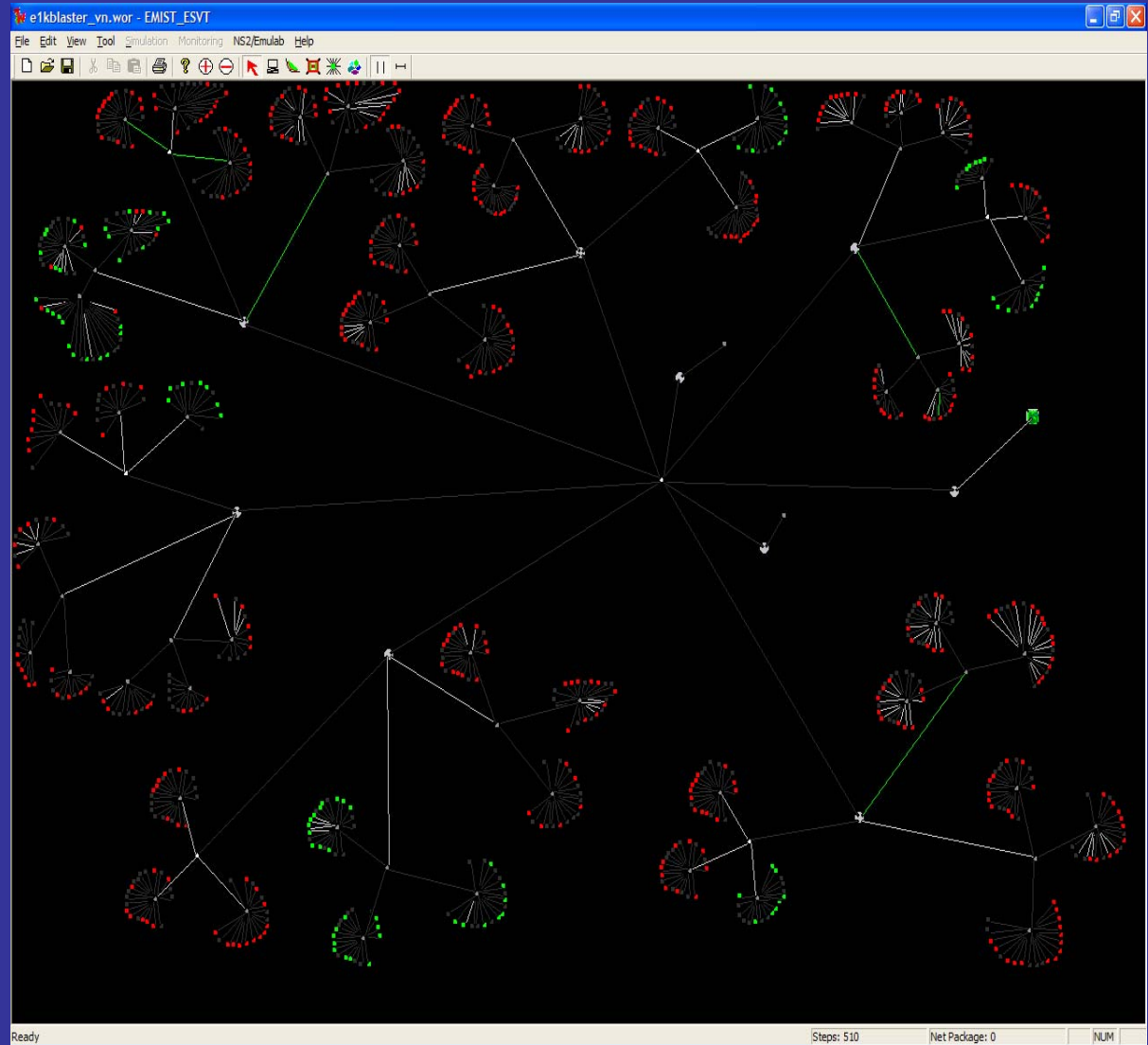
6.2. Placement of Honeypots



- Node 942 and 955 are two /24 honeypots
- Background traffic generators will skip these addresses
- Not effective for local-scanning worm if no traffic re-direction

ESVT Visualization

- Host state change
- Link color change (traffic)
- Link traffic bar chart
- Pause; Rewind; Drag
- Other advanced traffic chart



Future Study

- Topological worms
- Worm's impact on various services in the enterprise network
- Worm detection, containment, etc.

Questions?

Visit: <http://emist.ist.psu.edu>