

# Early Detection and Containment of Worm Epidemics

Tom Chen  
SMU, Dept of Electrical Engineering  
Dallas, Texas 75275  
[tchen@engr.smu.edu](mailto:tchen@engr.smu.edu)  
[www.engr.smu.edu/~tchen](http://www.engr.smu.edu/~tchen)

# Early Detection Systems

---

- Worm outbreaks can spread quickly, e.g., Slammer
- Early detection and warning systems correlate observations from distributed sensors to automatically detect new worm outbreaks, even unknown worms
  - Symantec's DeepSight Threat Management System
  - Internet Storm Center operated by SANS and Incidents.org
- Worm detection depends mostly on signatures

# Early Detection and Containment

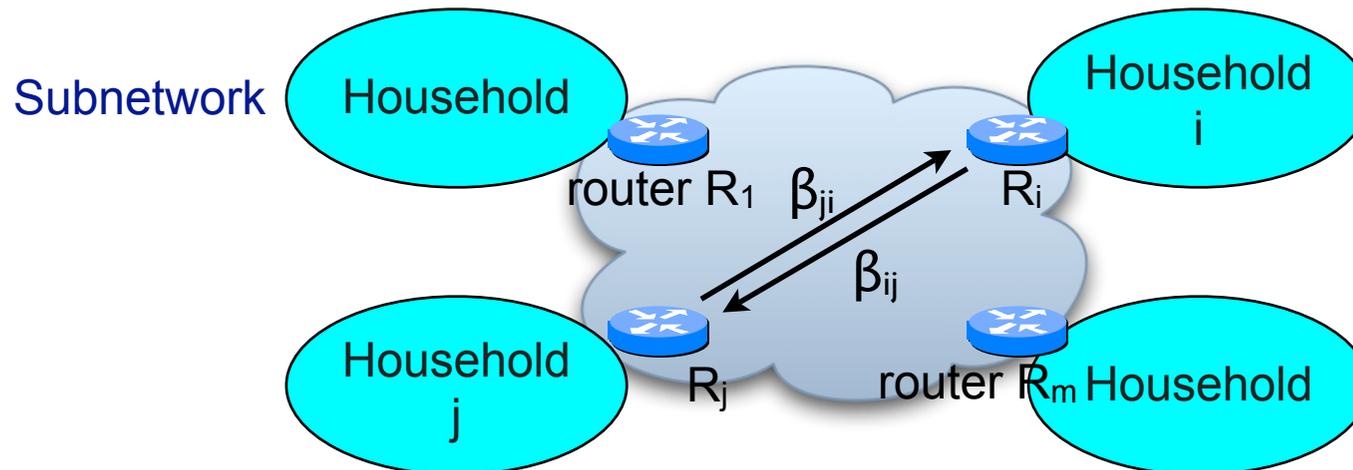
---

- Signatures allow accurate detection but
  - Worms without signatures may evade detection
  - Signatures for a new worm can take hours to develop
- Behavior-based (anomaly) detection is useful for detecting unknown worms
- After detection, outbreaks can be contained by quarantine (blocking) or rate throttling (slowing down)
- Outbreak behavior can be studied by epidemic modeling and simulation

# Community-of-Households Model

---

- Population consists of  $m$  households (autonomous systems)
  - Hosts are initially *susceptible* (S) state, then change to *infective* (I) state and *removed* (R) state
  - $\beta_{ij}$  = infectious contact rate from household  $i$  to household  $j$  (different than intra-household rates)



# Web-based Simulator

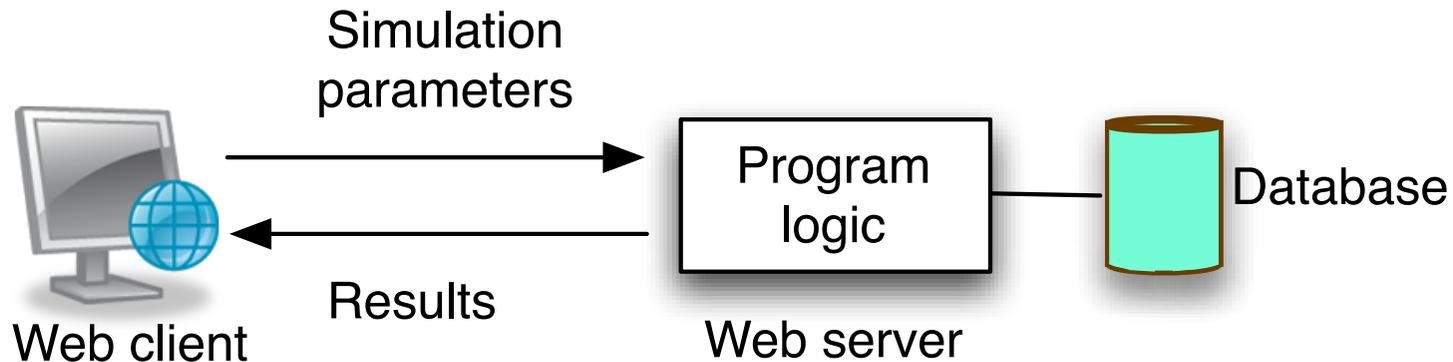
---

- In addition to mathematical analysis, we are developing Web-based worm simulator
  - Several existing worm simulators are written as applications, requiring users to download and compile
  - Simulators are platform dependent
  - Each copy of simulator and simulation results are tied to a physical machine
  - Users are responsible for maintaining and updating

# Web-based Simulator

---

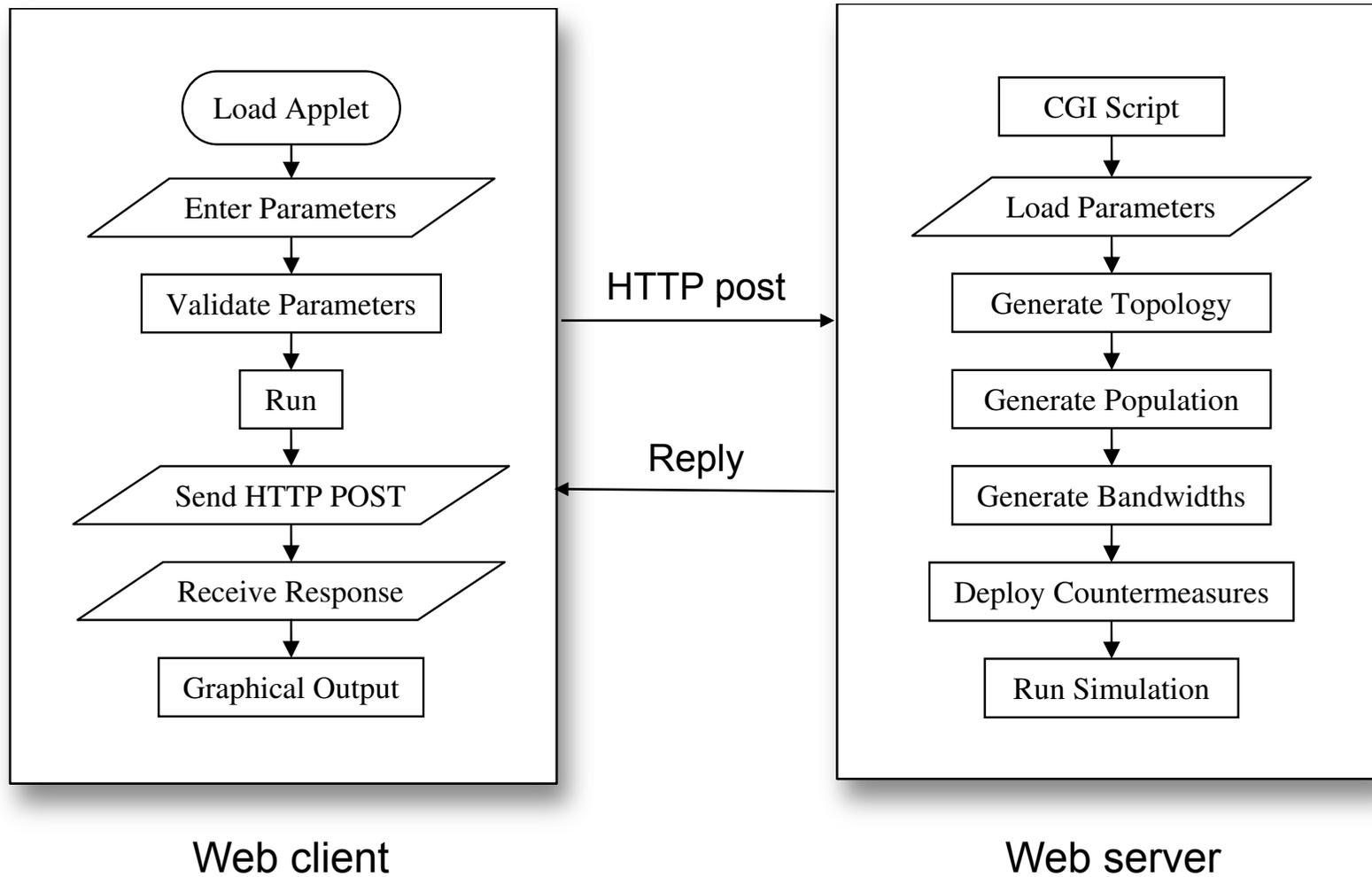
Client-server architecture separates GUI from program logic



- Web browser provides familiar, consistent, user-friendly GUI
- Users do not have to download and maintain their own simulators

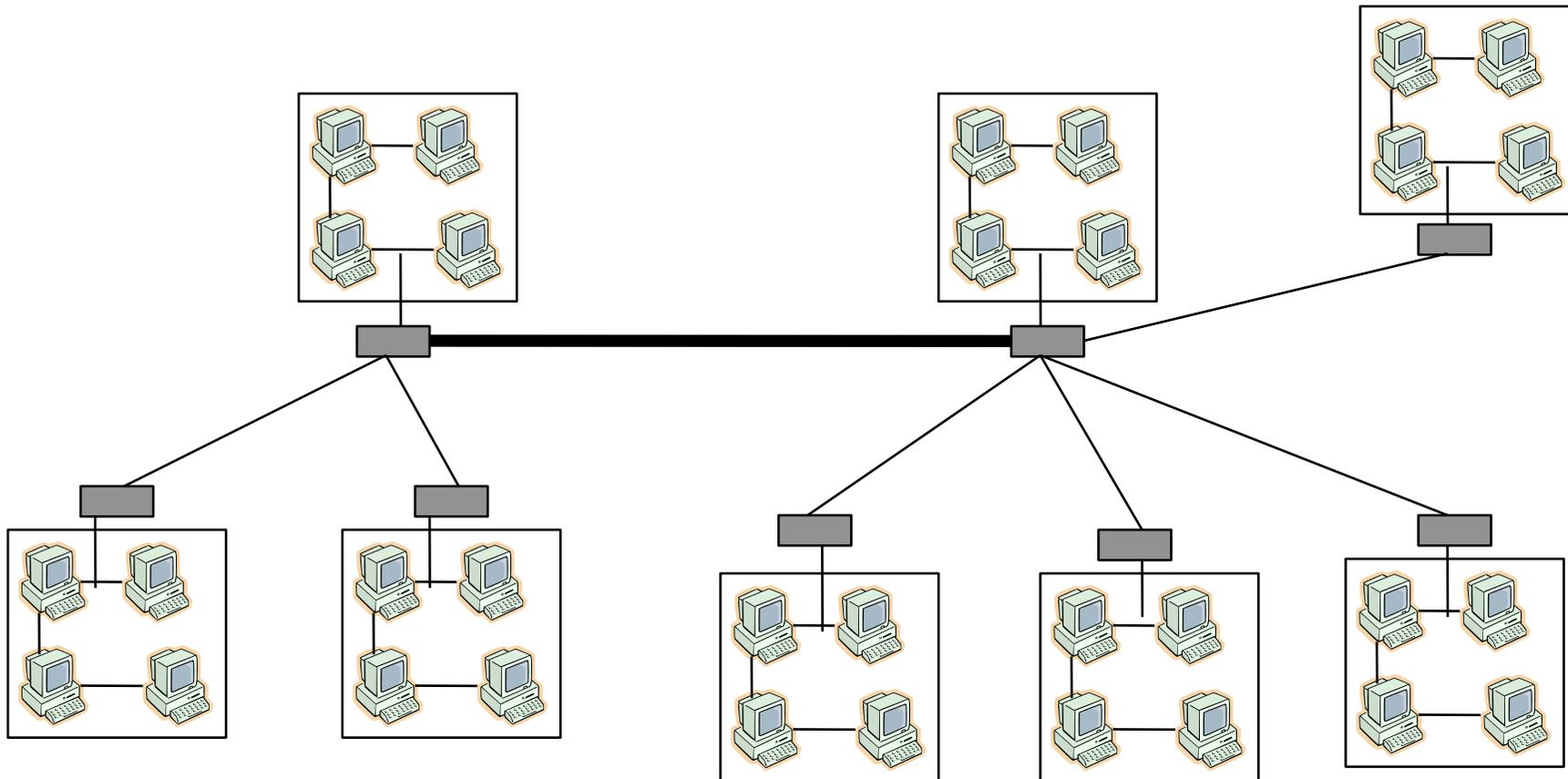
- Web server provides location-independent and platform-independent simulation
- Simulation results can be shared easily

# High Level Design



# Simulated Network Example

---



# Prototype GUI

The screenshot shows the NaSim Worm Simulator interface. On the left, a plot displays the number of infectives over time, with a y-axis labeled 'No. of Infectives' ranging from 0 to 16380 and an x-axis labeled 'Iterations' ranging from 0 to 171. The plot shows a sigmoidal curve. To the left of the plot are two vertical sliders. Below the plot is a horizontal slider and a status window showing 'Ready'. On the right, a control panel is divided into sections: 'Basic Parameters' (with fields for Num Houses, Population Uniform, Hosts / House, LinkCap Constt, Queue Len / Deg, Worm Spread Uniform, and Local Preference), 'Outbound Rate Control' (with an 'Activate' checkbox and fields for % age of Houses, Deployment Uniform, and Throttle Factor), 'Inbound Rate Control' (with an 'Activate' checkbox and fields for % age of Houses, Deployment Uniform, Trigger Global, Threshold, and Throttle Factor), and 'Quarantining' (with an 'Activate' checkbox and fields for % age of Houses, Deployment Uniform, Trigger Global, and Threshold). A 'Run' button is at the bottom of the control panel. Annotations with blue arrows point to various elements: 'Plot of infections per time' points to the graph; 'Sliders to adjust plot' points to the vertical sliders; 'Status and help window' points to the 'Ready' status box; 'Model parameters' points to the 'Basic Parameters' section; 'Rate throttling parameters' points to the 'Outbound Rate Control' and 'Inbound Rate Control' sections; and 'Quarantine parameters' points to the 'Quarantining' section.

Plot of infections per time

Sliders to adjust plot

Status and help window

Model parameters

Rate throttling parameters

Quarantine parameters

# Prototype Current Features

---

- GUI is Java applet for data entry, data validation, context-based help, graphical display of results
- CGI script is python program to pass input parameters to core simulator program running on server back end
- Core simulator uses U. Michigan's Inet 3.0 to generate network topology
- Simulation results are stored on server (with unique identifiers) for later retrieval or sharing

# Issues and Future Features

---

- Topology generation and simulation time slows down drastically with model size (number of households)
- Server can keep track of multiple simultaneous simulations (by job scheduling) but number is currently limited to prevent overwhelming
- Currently static routing (shortest routes computed by Dijkstra's algorithm) but dynamic routing more realistic
- Rate throttling not fully implemented yet