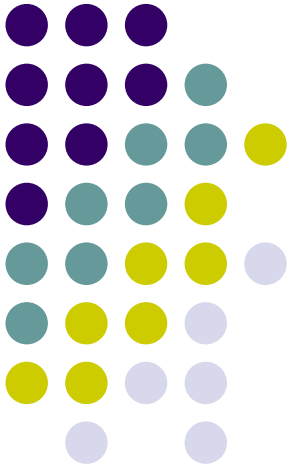
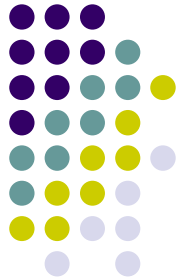
Sencun Zhu
Assistant Professor
CSE and IST, PSU

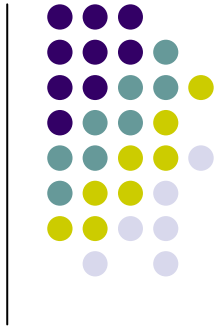


Research



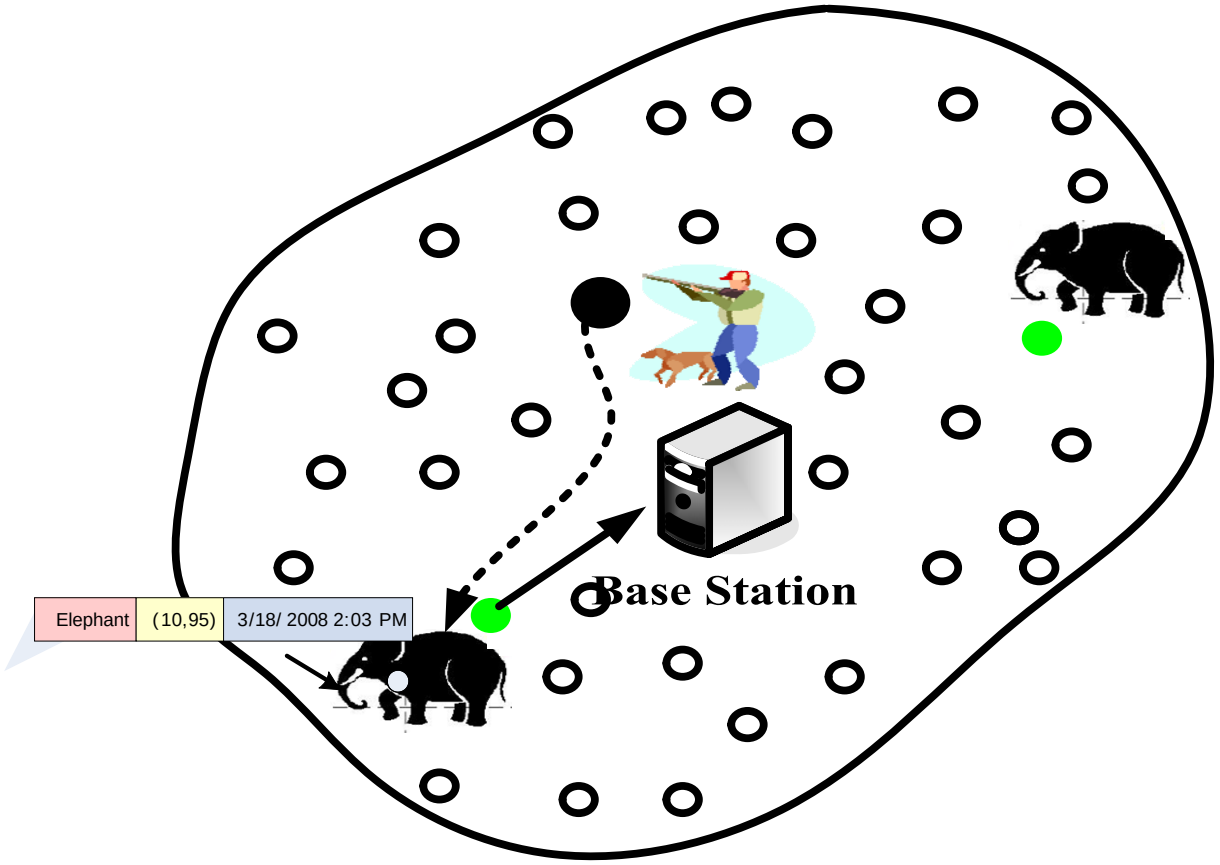
- Interest
  - Security, networking
- Recent professional activities
  - Program Co-Chair: ACM SASN'06.
  - TPC member: ACM CCS'07, IEEE Infocom'07, ESORICS'08, ACM WiSec'09, IEEE ICDCS'09.
  - Treasurer: ACM CCS'07, 08, 09
- Research supports
  - Army Research Office (ARO), NSF CyberTrust, TTC

Current Projects

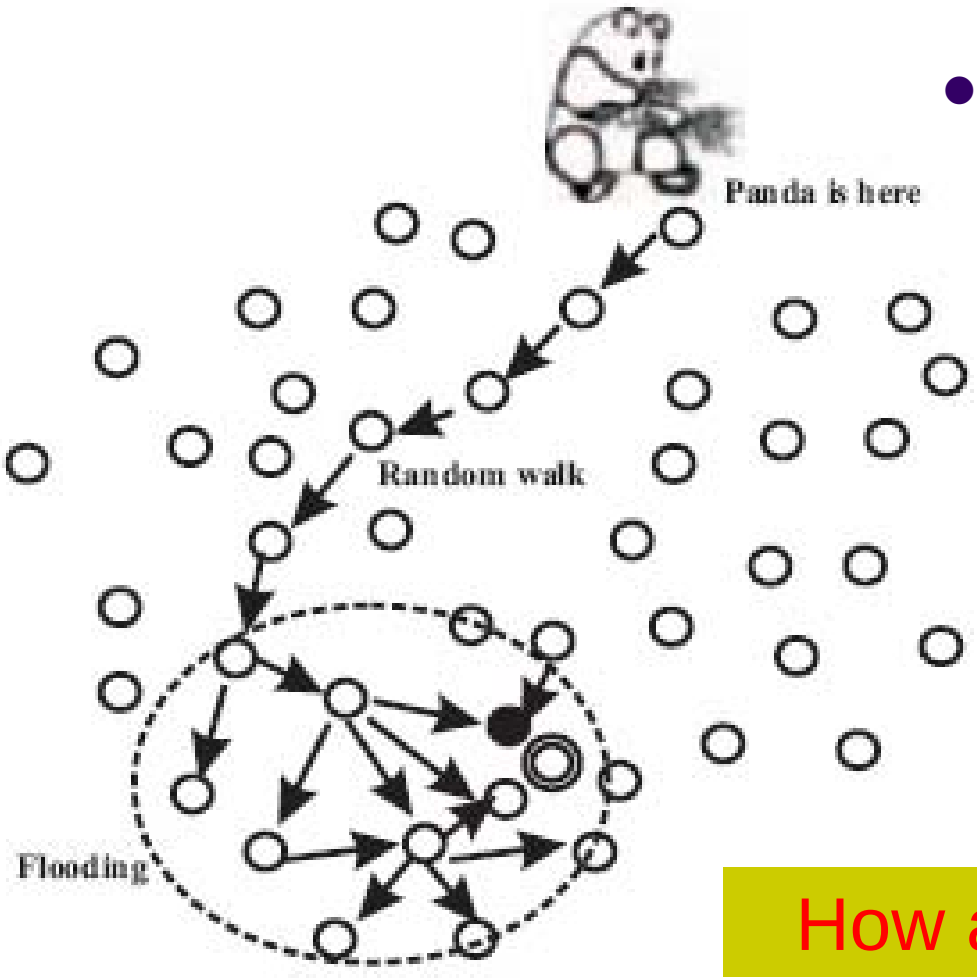


- Security and privacy for sensor networks
  - Source location anonymity under different attack models
  - Applications of sensor networks to public safety
  - Sensor worm containment via software diversity
  - Defending against channel jamming and interference
- Security for MANET/DTN
  - Traceback in mobile ad hoc networks
  - Broadcast authentication in DTN
  - DoS detection and robust forwarding in DTN
- Security for Peer-to-Peer Networks
  - Pollution attack defense
  - P2P worm containment
- Security for cellular networks
  - Cellphone worm/malware detection and containment
- Code security
  - Blocking buffer overflow attacks by static code analysis
  - Code plagiarism detection

Source Location Anonymity



Panda-Hunter Game [SASN'04]

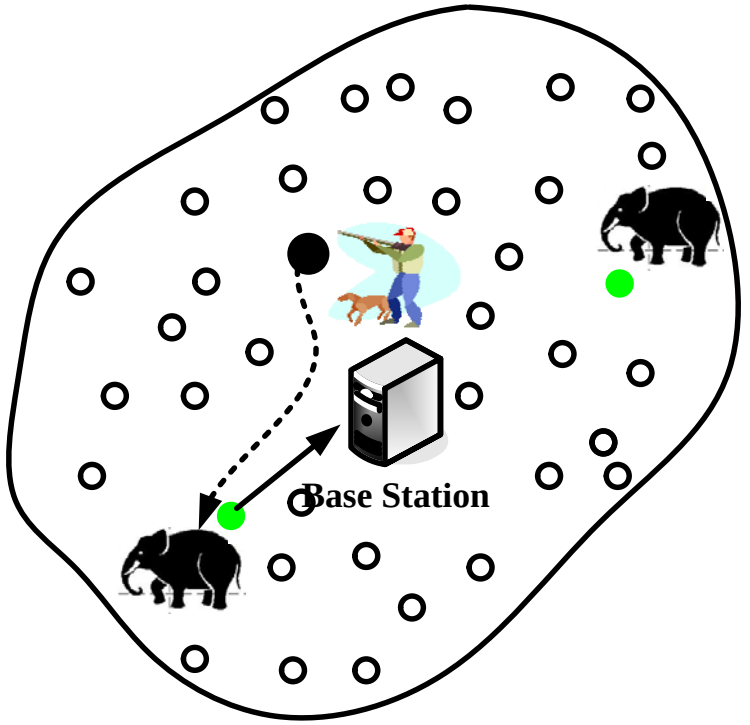


- Phantom Routing:
  - Local attackers
  - Capture likelihood is 97%, if attacker's hearing range $\geqslant$ 3 times of that of sensors

How about global attackers?

Key Observation:

- Dummy traffic is needed for event unobservability under a global attack model

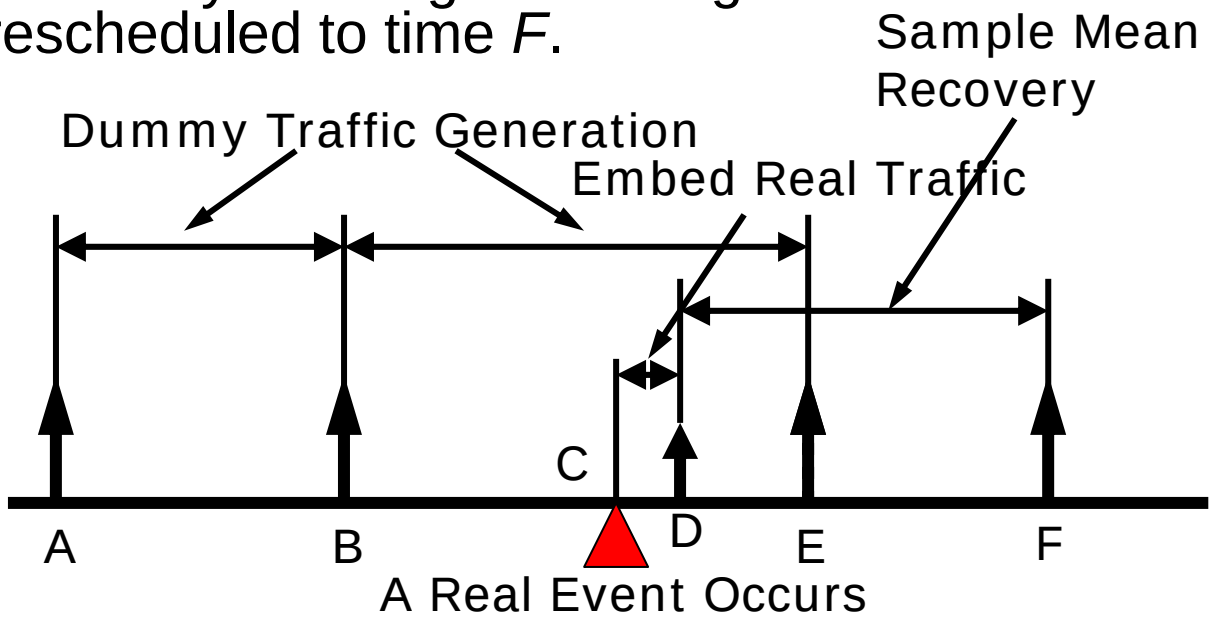


Research Issues:

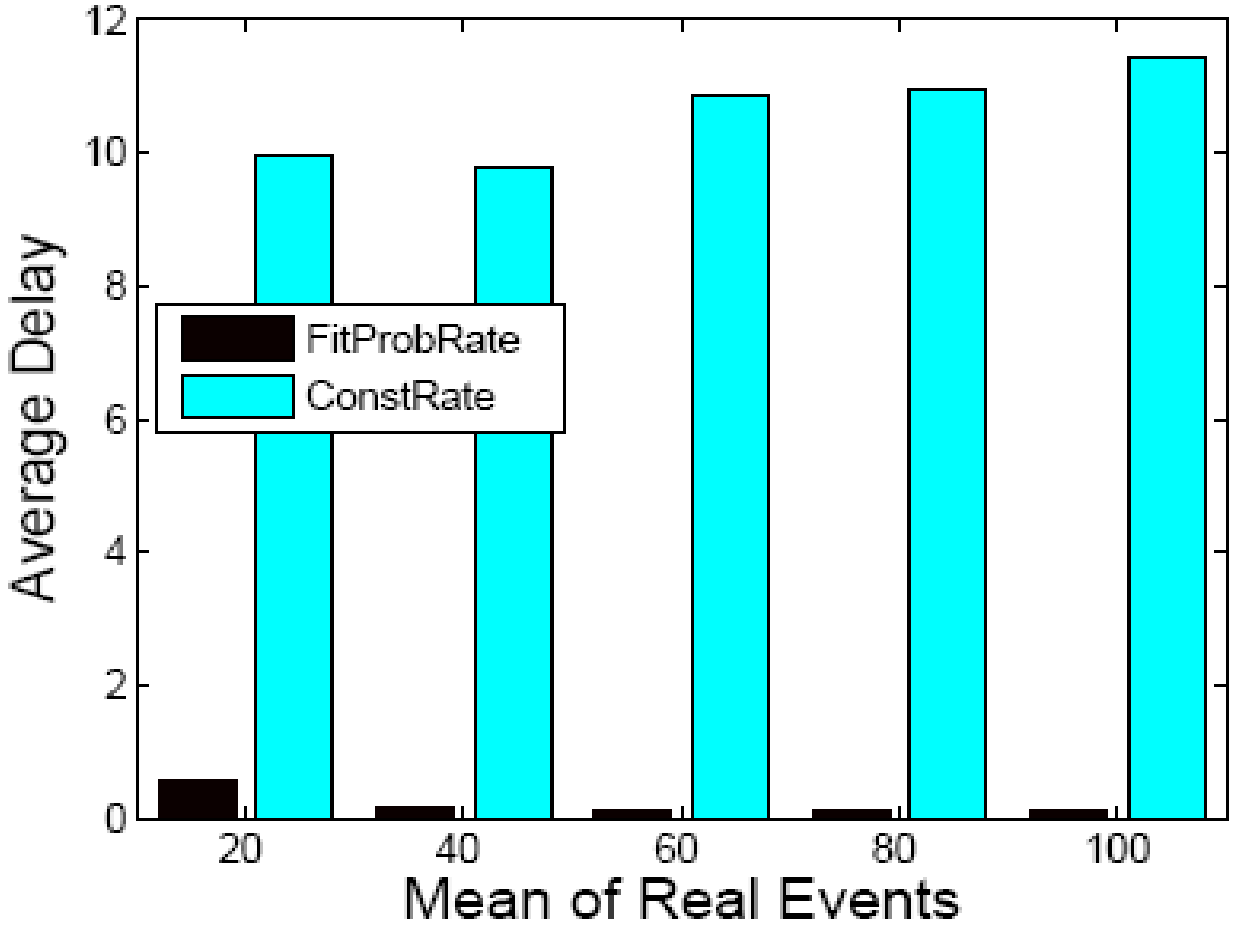
- How to generate dummy traffic?
- How to forward them?

A Running Example

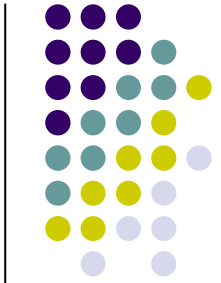
- Defender:
  - Dummy Traffic Generation: three bogus messages are supposed to be generated at A, B, E;
  - A real event occurs at C
  - Embed Real Traffic: the real event is sent out at D
  - Sample Mean Recovery: the bogus message at time *E* is cancelled and rescheduled to time *F*.
- Attacker:
  - Attackers see the intervals between
    - *A* and *B*
    - *B* and *D*
    - *D* and *F*



Average Latency Comparison



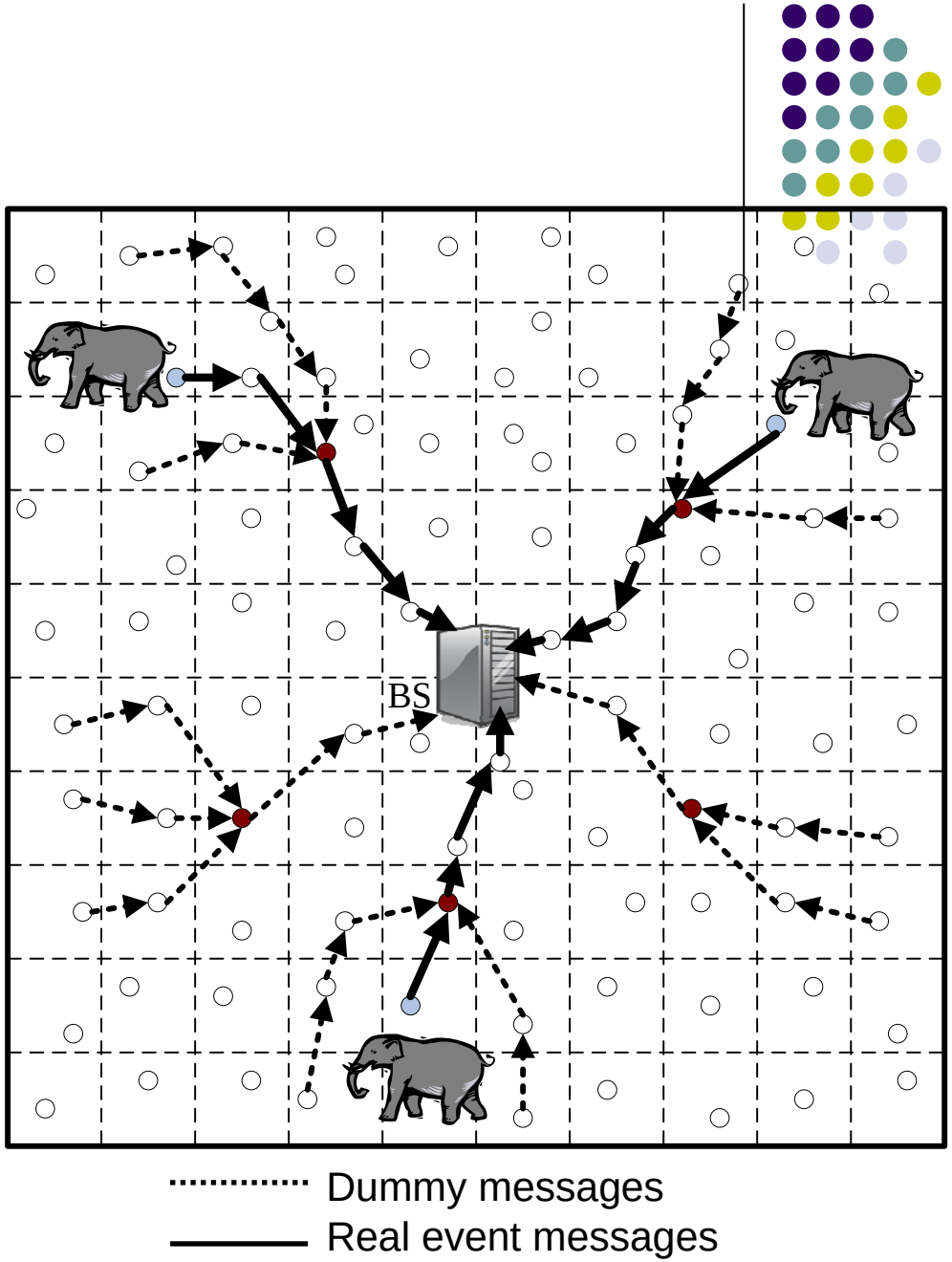
Two Filtering schemes



- Proxy-based Filtering Scheme (PFS)
  - Proxy nodes collect and filter en-route dummy traffic
- Tree-based Filtering Scheme (TFS)
  - Proxy hierarchy to further reduce message overhead by multiple layers of filtering

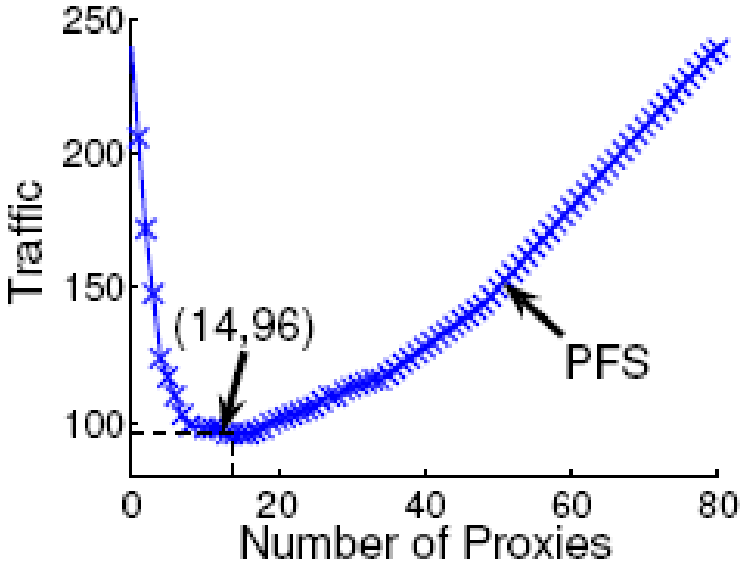
PFS (1)

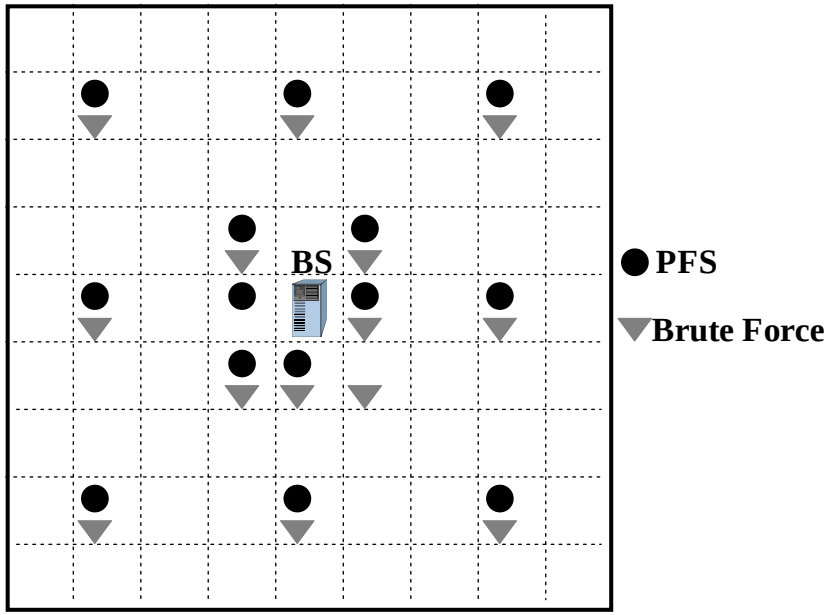
- How many proxies needed?
- How to place proxies to minimize network traffic?
- What is filtering operation inside each proxy?



PFS (2)

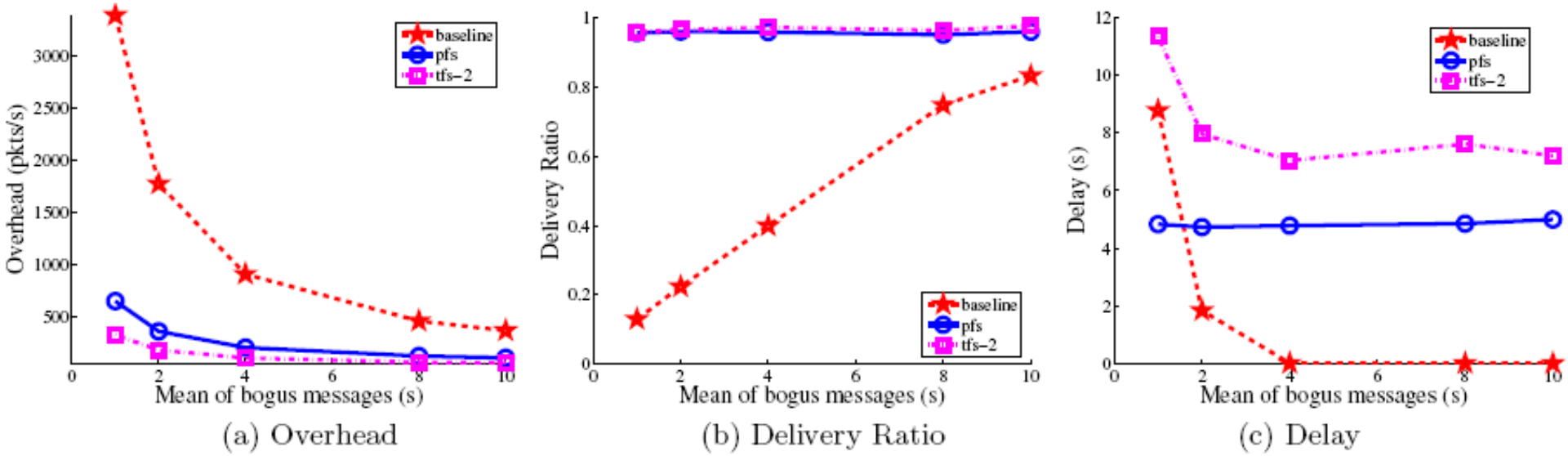
- Optimal proxy placement algorithm
  - To minimize overhead $r_{source} \times \sum_{i \in V} \min_{j \in P} d(i,j) + r_{proxy} \times \sum_{j \in P} c(j)$
    - Modeling of the optimization: *facility location problem* (NP-hard)
    - *Local search heuristics* to find an optimal solution (proxy number and placement) in a more efficient way



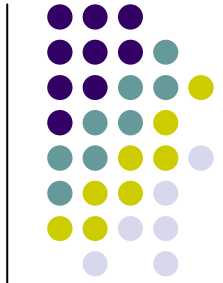


Performance Evaluation

- Simulation results under high-rate real events



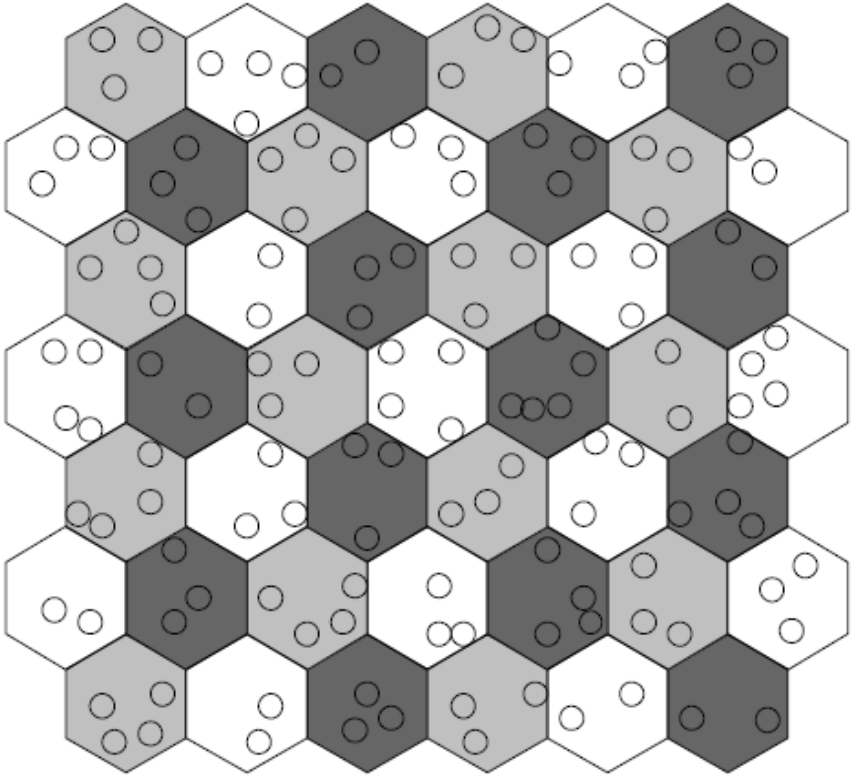
Sensor Worm Contaiment



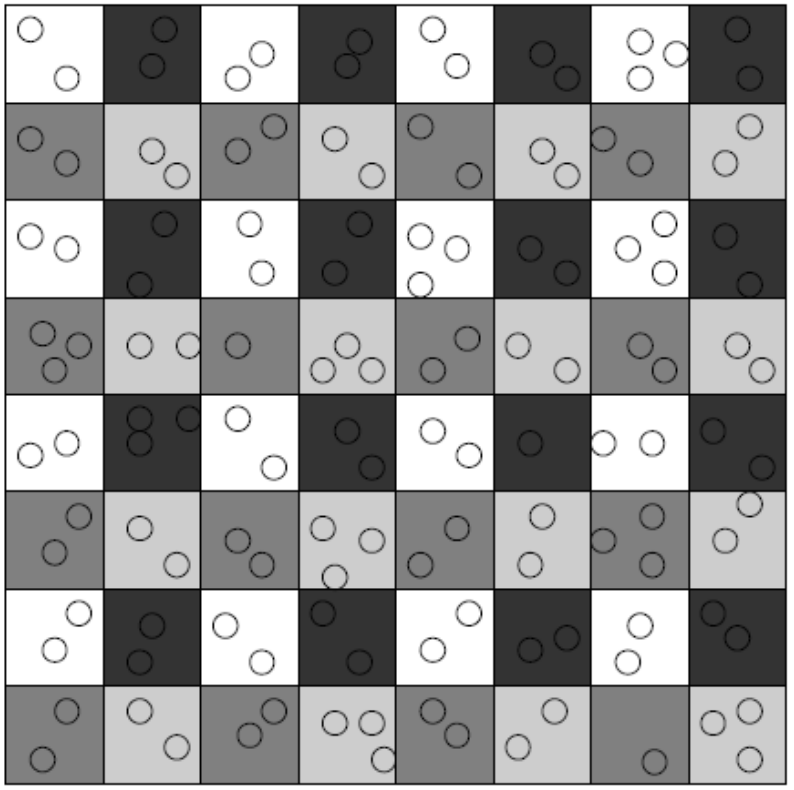
- Contribution I:
  - We illustrate the feasibility of sensor worms through trial experiments on Mica2 motes
- Contribution II:
  - In spirit of *survivability through heterogeneity* philosophy, we explore *software diversity* for sensor worm defense
    - Graph construction
    - Program assignment algorithm
    - Analysis on impact of deployment error to worm propagation
    - Simulation results to validate effectiveness of proposed scheme

Sensor Worm Defense

- Our solution
  - Partition the sensor field into cells, and assign a color to a cell



(a) 3-color case

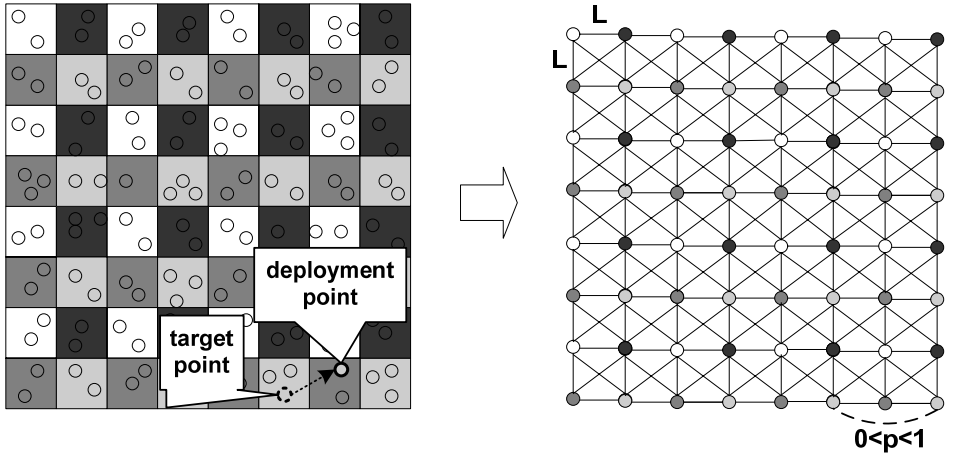


(b) 4-color case

The Impact of sensor deployment error

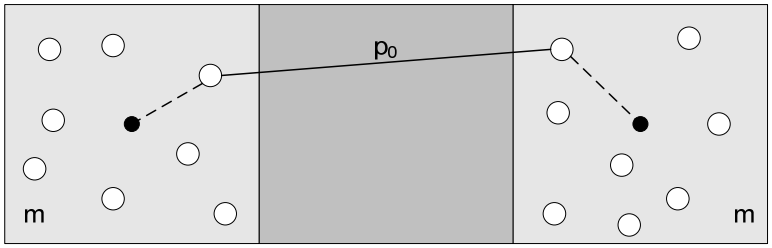
- *Deployment points* are modeled by *two-dimensional normal distribution* with *target points* as mean
- p_0: prob. that two nodes from neighboring cells with same color are connected

$$p_0 = \int_{x=0}^{X} \int_{y=0}^{Y} \frac{p_{n2}}{2\pi\sigma^2} e^{-[(x-x_1)^2+(y-y_1)^2]/2\sigma^2} dxdy$$

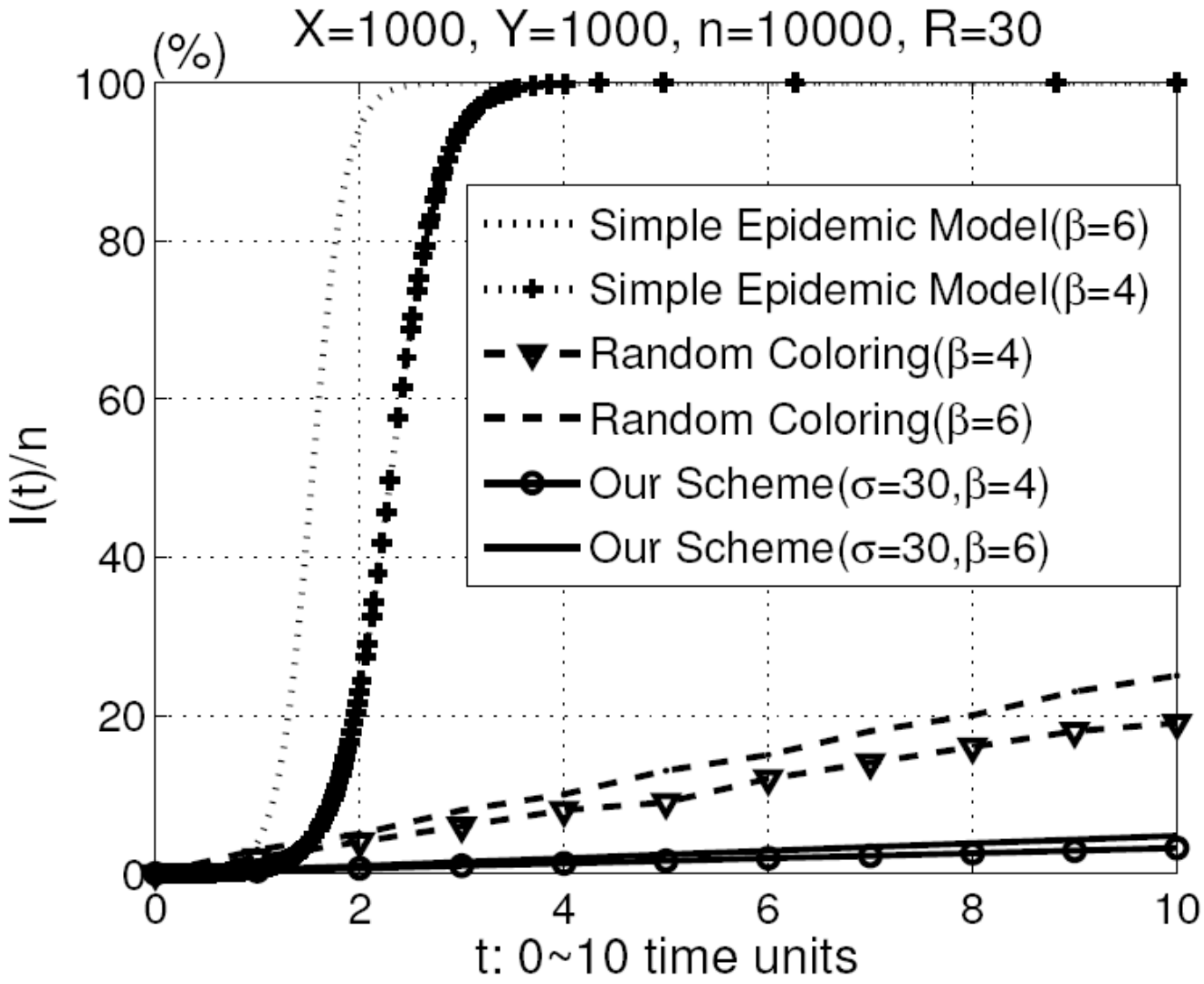


- p: prob. that two neighboring cells with same color are connected

$$p = 1-(1-p_0)^{m^2}$$



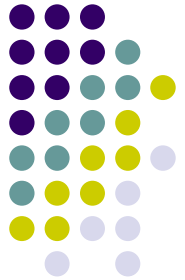
Comparison


X=1000, Y=1000, n=10000, R=30
(%)
100
80
60
40
20
0
I(t)/n
0
2
4
6
8
10
t: 0~10 time units
Simple Epidemic Model(β=6)
Simple Epidemic Model(β=4)
Random Coloring(β=4)
Random Coloring(β=6)
Our Scheme(σ=30,β=4)
Our Scheme(σ=30,β=6)

Remote Code Execution

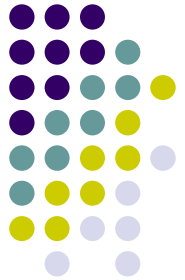
- Most of security attacks/vulnerabilities reported in CERT and Microsoft Security Bulletin are buffer-overflow based remote code execution
- It allows attackers to use a network request to inject a piece of exploit code into the “body" of a service or application program
- It is a root cause for most of the cyber attacks such as server breaking-in, worms, zombies, and botnets.

Obfuscation of Exploit Code



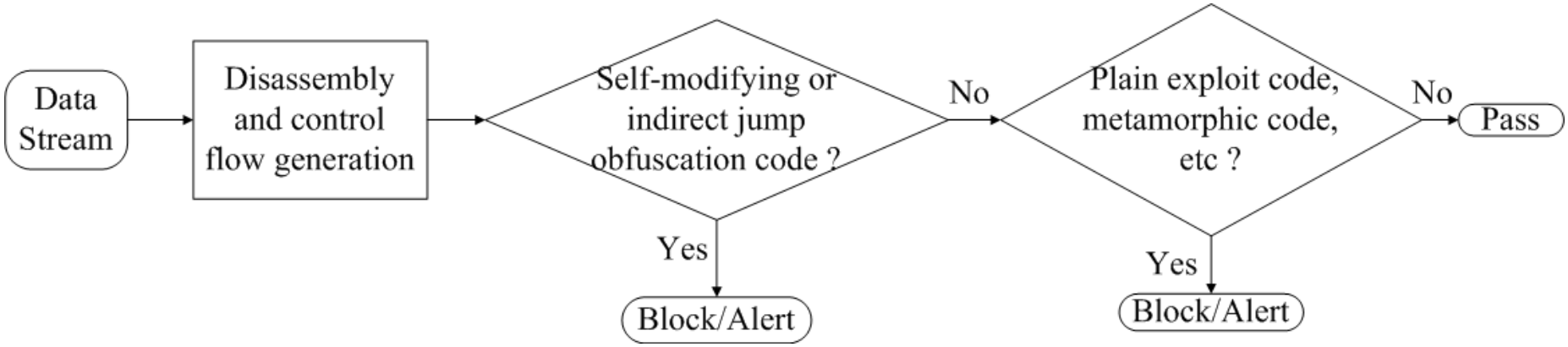
- Anti-signature
  - Polymorphism
  - Metamorphism (instruction reordering, garbage insertion, instruction replacement,...)
- Anti-static-analysis
  - Self-Modifying
  - Anti-disassembly (junk byte insertion, indirect jump, branch function, …)
- Anti-emulation (inserting delay loops, executing random code, …)

Design Goal



- Robust to (almost) all anti-signature, anti-static-analysis and anti-emulation obfuscation
  - requirement: detecting not only self-modifying code but also other types of attack payload
- Blocking new and unknown remote code injection attacks (e.g., zero-day exploit code)
  - requirement: signature free

An Overview of STILL



Enabling Techniques

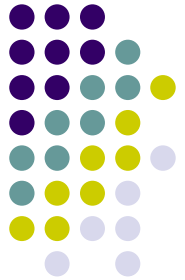
- Static Taint Analysis
- Initialization Analysis

Evaluation- Detection Effectiveness

- Test Data Set
  - 12,000 polymorphic attack messages from 10 publicly available polymorphic engines, all of which encrypt the original shellcode
  - 23 different Windows plain shellcode and 11 different Linux plain shellcode in Metasploit framework
  - Worms: CodeRed I, CodeRed II, Sasser, Slammer and Blaster

- STILL detects all the above attacks

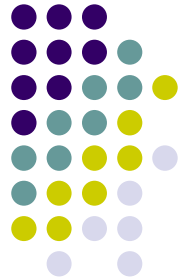
Comparison with Snort



•475,297 incoming UDP/TCP packets collected at a honeypot platform in France (Leurrecom) from 03/22/2007 to 04/21/2007

	Numbers of Attacks
Detected by both STILL and Snort	**2,029**
Detected only by Snort	**15 (2 FP,13 directory traversal attacks)**
Detected only by STILL	**45 (6 polymorphic code, 39 plain code)**

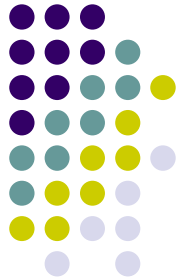
Normal Data Sets for False Positive Testing



Name	Requests	Replies	Size(MB)
Dataset1	5,026	3,584	317.24
Dataset2	32,666	27,253	288.53
Dataset3	87,260	52,772	355.99
Dataset4	17,871	16,941	320.91
Dataset5	26,759	28,028	129.41
Dataset6	17,075	10,042	237.52
Dataset7	46,107	6,774	158.28
Total	**232,764**	**145,394**	**1,807.88**

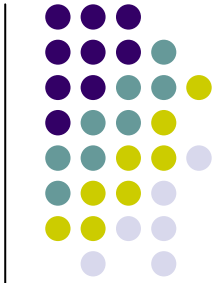
- Real HTTP traffic captured in local network of security lab of Penn State. The replies and requests include various types of multimedia. The overall size of the traces is about 1.77 GB

False Positive



Mime-type	Warning Type			
	1	2	3	4
application/octetstream	10	1	0	0
application/x-mms-framed	0	1	0	0
application/x-shockwave-flash	0	2	0	0
audio/mpeg	0	1	0	0
image/gif	0	6	0	0
image/jpeg	0	2	0	0
text/plain	0	1	0	0
flv-application/octet-stream	0	0	0	2
video/flv	0	0	0	1
video/x-flv	0	1	0	2
Total	**10**	**15**	**0**	**5**

- (Type1) A function call exploit code is detected
- (Type2) A system call exploit code is detected
- (Type3) An indirect jump obfuscation code is detected
- (Type4) A self-modifying obfuscation code is detected. Note that polymorphic exploit code is included in Type4



Thanks!

szhu@cse.psu.edu
www.cse.psu.edu/~szhu