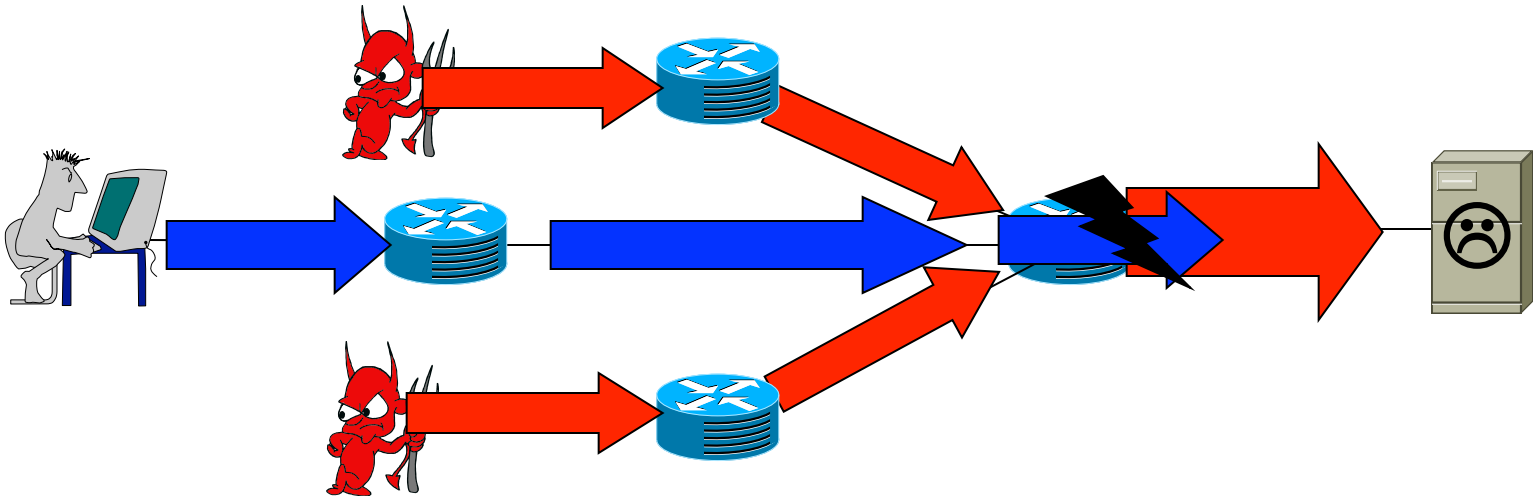
CPS 590.5 Computer Security
Lecture 10: IP Traceback

Xiaowei Yang
xwy@cs.duke.edu

Roadmap

- Previous lecture
  - Practical network security
- Today
  - IP traceback
    - Packet state based
    - Router state based

How to Attack : Exhausting shared resources

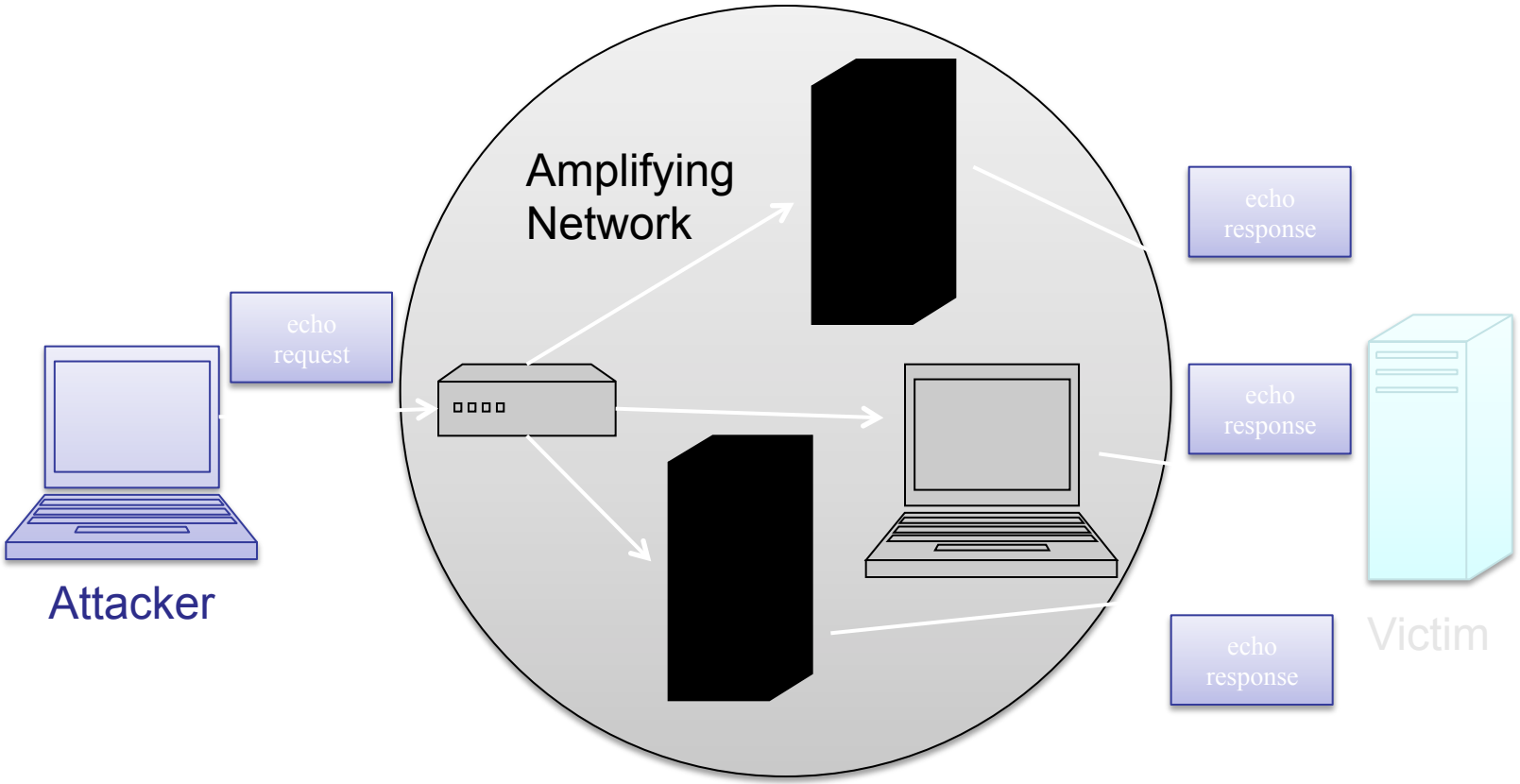


- Flooding traffic to exhaust the bandwidth, memory, or CPU of a victim
  - Spoof addresses to hide
    - Passport
  - Distributed DoS (DDoS) to hide and to maximize damage
    - Multiple (weak) machines against (strong) victim

IP address spoofing

- IP source address can be spoofed
- Challenges
  - No accountability
  - Filters do not work well
  - Reflector attacks

Smurf/Reflector Attack



Practical Network Support for IP Traceback

Stefan Savage University of Washington/
University of California, San Diego

David Wetherall, Anna Karlin and Tom Anderson University of Washington, Seattle

Single-Packet IP Traceback

Alex C. Snoeren

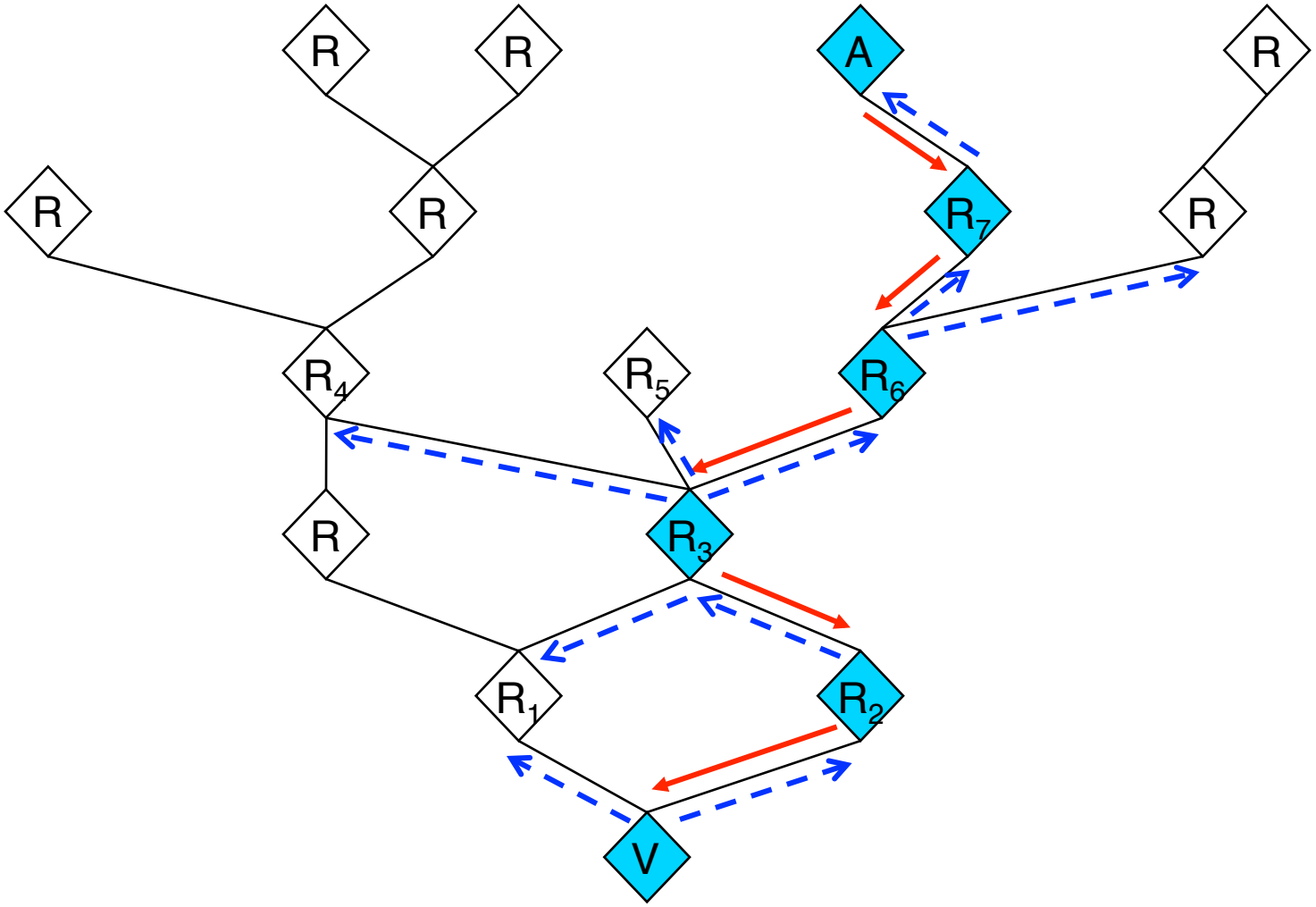
BBN Technologies

(with Craig Partridge, Tim Strayer, Christine Jones,
Fabrice Tchakountio, Beverly Schwartz, Matthew Condell,
Bob Clements, and Steve Kent)

Low-rate attacks

- Not all attacks are large flooding DOS attacks
- Well-placed single packet attacks
- Packets may have spoofed IP addresses
- How to track these attacks and find their origin?

IP Traceback



Logging Challenges

- Attack path reconstruction is difficult
  - Packet may be transformed as it moves through the network
- Full packet storage is problematic
  - Memory requirements are prohibitive at high line speeds (OC-192 is ~10Mpkt/sec)
- Extensive packet logs are a privacy risk
  - Traffic repositories may aid eavesdroppers

Single-Packet Traceback: Goals

- Trace a *single* IP packet back to source
  - Asymmetric attacks (*e.g.*, Fraggle, Teardrop, ping-of-death)

- Minimal cost (resource usage)

One solution: Source Path Isolation Engine (SPIE)

SPIE Architecture

- DGA: Data Generation Agent
  - computes and stores digests of each packet on forwarding path.
  - Deploy 1 DGA per router
- SCAR: SPIE Collection and Reduction agent
  - Long term storage for needed packet digests
  - Assembles attack graph for local topology
- STM: SPIE Traceback Manager
  - Interfaces with IDS
  - Verifies integrity and authenticity of Traceback call
  - Sends requests to SCAR for local graphs
  - Assembles attack graph from SCAR input

Data Generation Agents

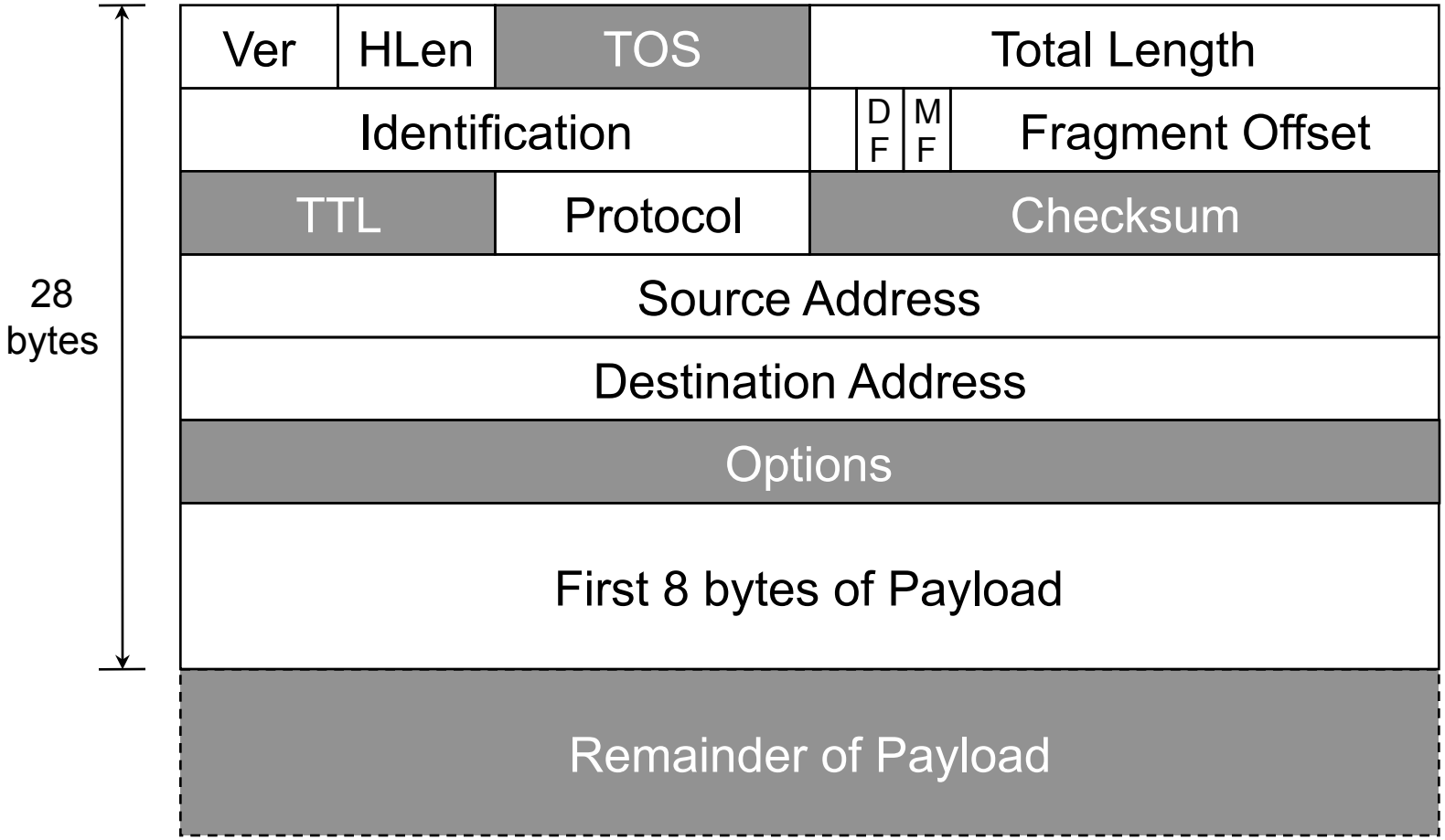
- Compute “packet digest”
- Store in Bloom filter
- Flush filter periodically

Packet Digests

- Compute hash(p)
  - Invariant fields of p only
  - 28 bytes hash input, 0.00092% WAN collision rate
  - Fixed sized hash output, *n*-bits

- Compute *k* independent digests
  - Increased robustness
  - Reduced collisions, reduced false positive rate

Hash input: Invariant Content



Hashing Properties

- Each hash function
  - Uniform distribution of input -> output

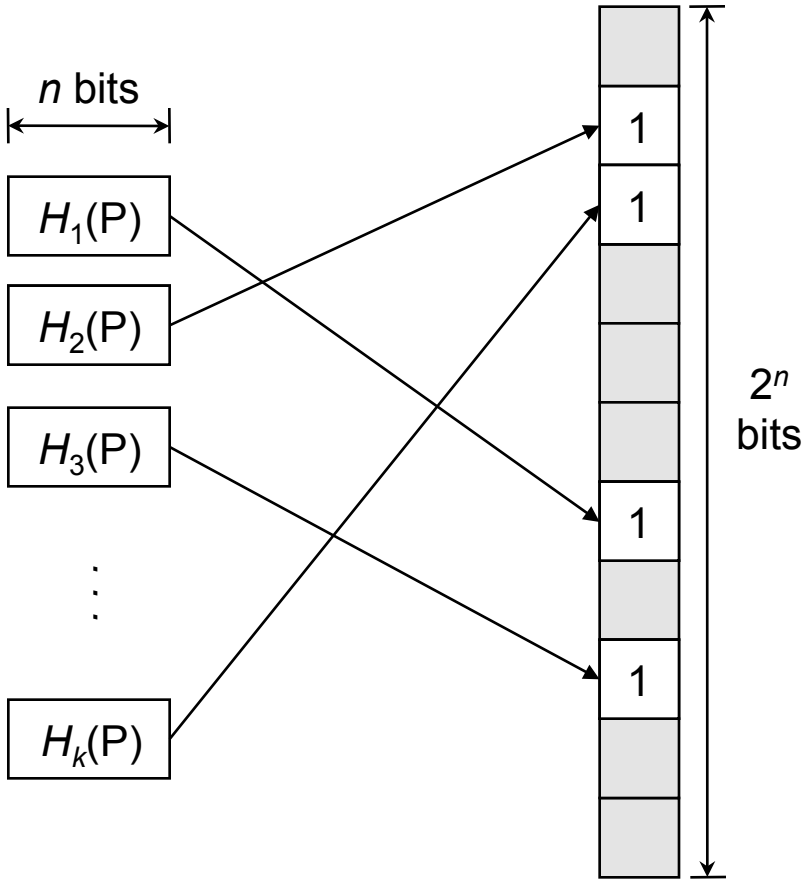
    H1(x) = H1(y) for some x,y -> unlikely

- Use k independent hash functions
  - Collisions among k functions independent
  - H1(x) = H2(y) for some x,y -> unlikely

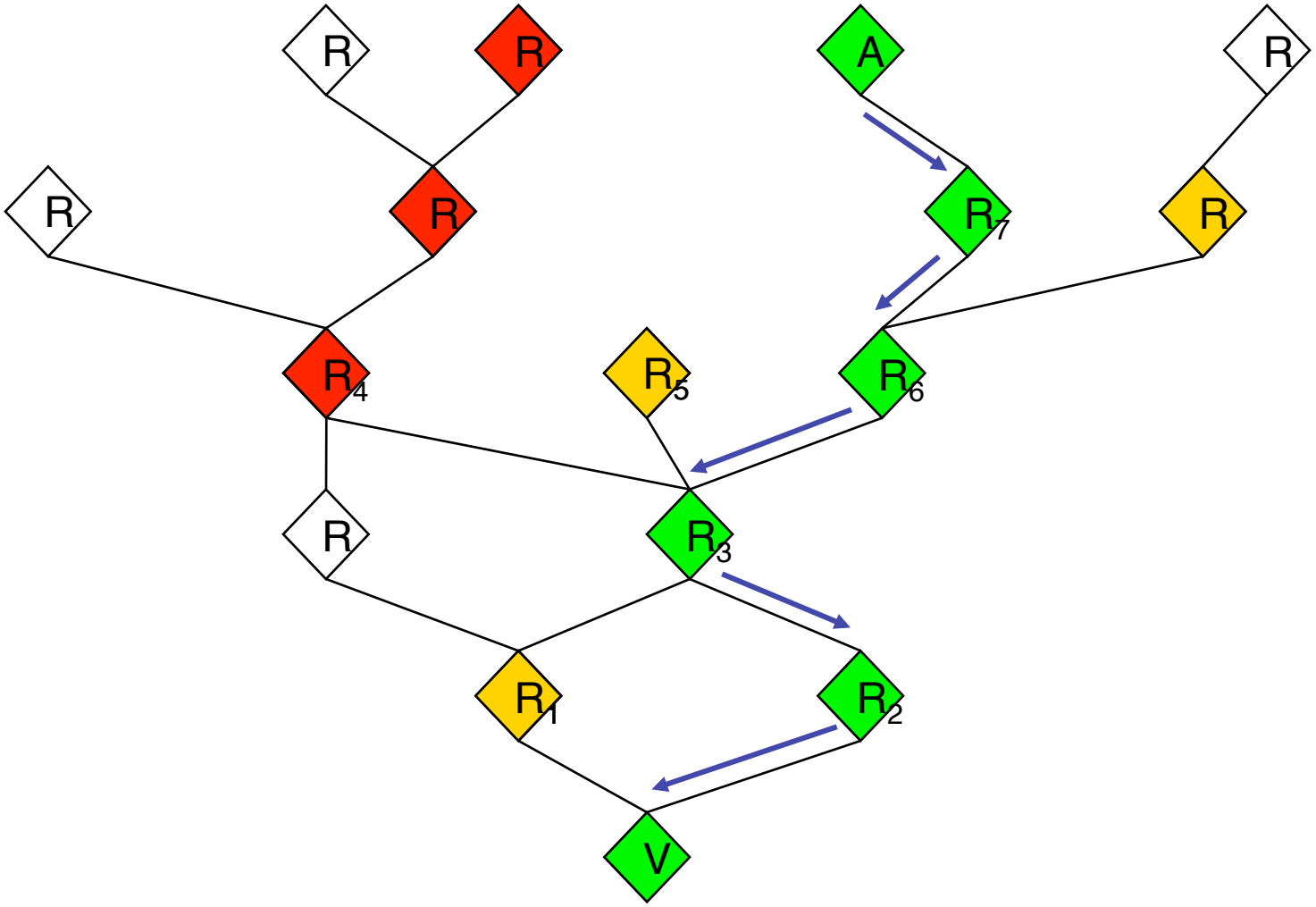
- Cycle k functions every time interval, t

Digest Storage: Bloom Filters

- **Fixed structure size**
  - Uses 2^n bit array
  - Initialized to zeros
- **Insertion**
  - Use *n*-bit digest as indices into bit array
  - Set to '1'
- **Membership**
  - Compute *k* digests, d_1, d_2, etc…
  - If (filter[d_i]=1) for all i, router forwarded packet



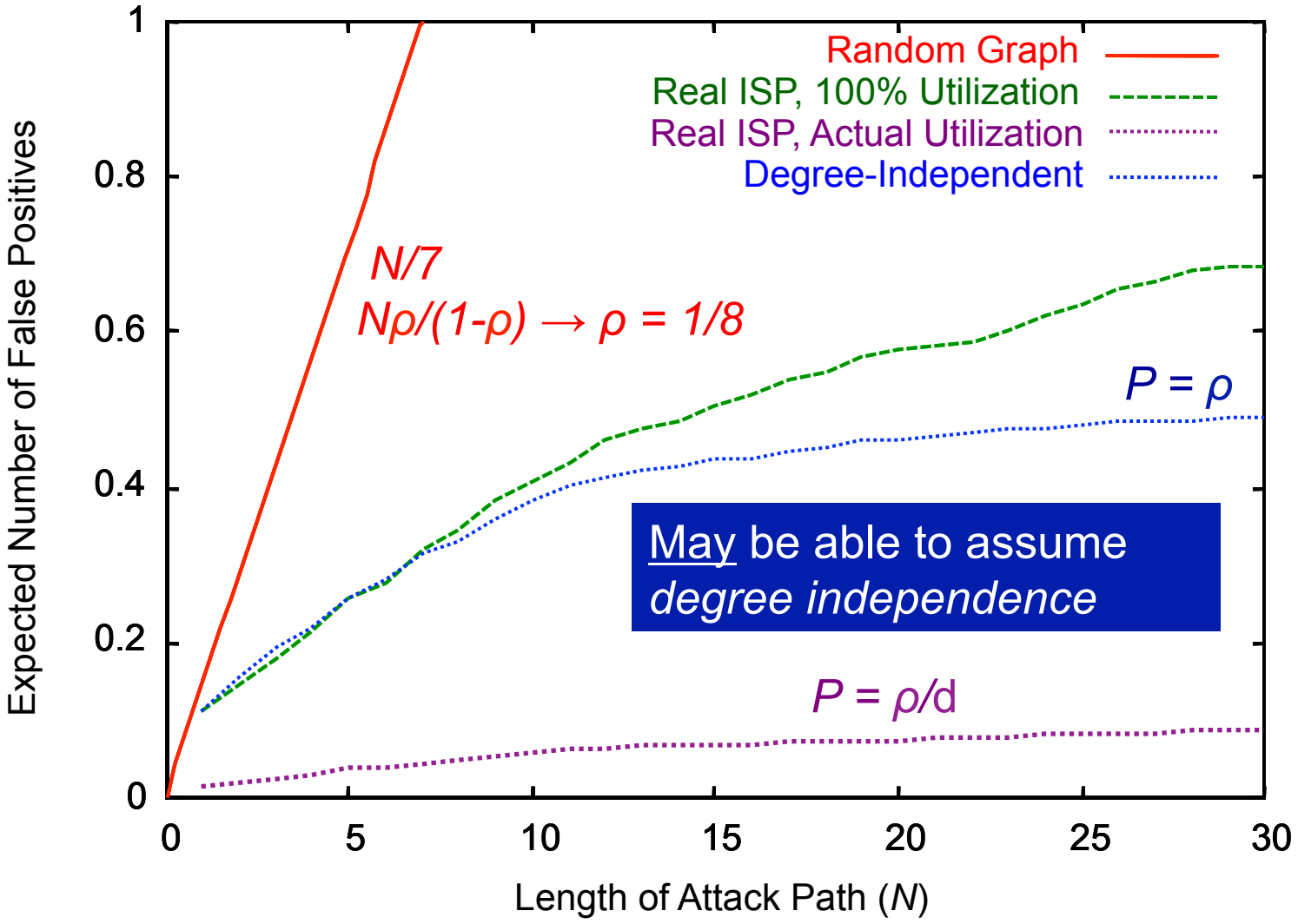
False Positive Distribution



Adjusting Graph Accuracy

- False positives rate depends on:
  - Length of the attack path, N
  - Complexity of network topology, d
  - Capacity of Bloom filters, P
- Bloom filter capacity is easy to adjust
  - Required filter capacity varies with router speed and number of neighbors
  - Appropriate capacity settings achieve linear error growth with path length

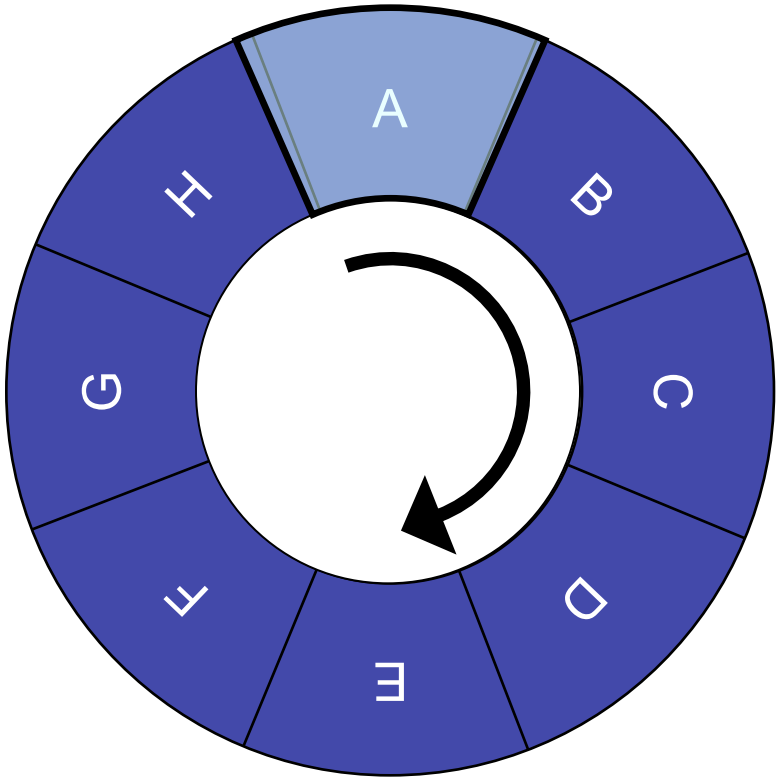
Simulation Results



How Big are Digests?

- Quick rule of thumb:
  - $\rho = 1/8$, assuming degree independence
  - Bloom filter $k = 3$, $M/n =$ *5 bits per packet.*
  - Assume packets are ~1000 bits
- Filters require *~0.5%* of link capacity
  - Four OC-3s require 47MB per minute
  - 128 OC-192 links need <100GB per minute
- Access times are equally important
  - Current drives can write >3GB per minute
  - OC-192 needs SRAM access times

Filter Paging



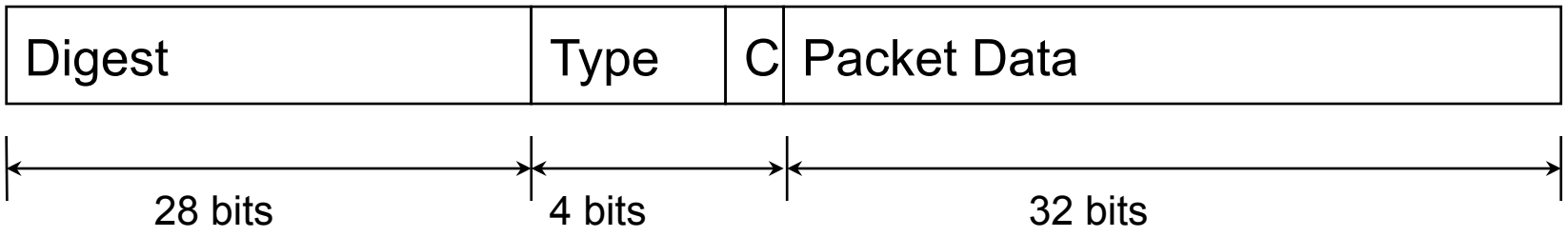
- "Small" Bloom filters
  - Random access
  - Need fast memory
- Store multiple filters
  - Increase time span
  - Ring buffer avoids memory copies
- Timestamp each bin
  - Fence-post issues

Transformations

- Occasionally invariant content changes
  - Network Address Translation (NAT)
  - IP/IPsec Encapsulation, etc.
  - IP Fragmentation
  - ICMP errors/requests
- Routers need to invert these transforms
  - Often requires additional information
  - Can store this information at the router

Transform Lookup Table

- Only need to restore invariant content
  - Often available from the transform (*e.g.*, ICMP)
- Otherwise, save data at transforming router
  - Index required data by transformed packet digest
  - Record transform type and sufficient data to invert
- Bounded by transform performance of router

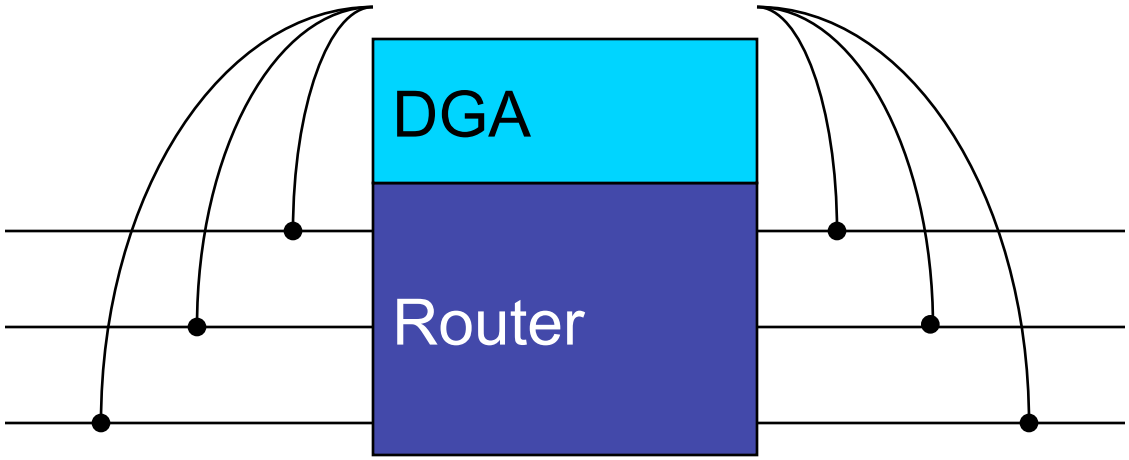


Prototype Implementation

- Implemented in PC-based routers
  - Both FreeBSD and Linux implementations
    - Packet digesting on kernel forwarding path
  - Zero-copy kernel/user digest tables
    - Digest tables and TLT stored in kernel space
- User-level query-support daemons
  - Supports automatic topology discovery
  - Queries automatically triggered by IDS

SPIEDER Approach

Each router has an *external* Data Generation Agent (DGA)



SPIE DGA Encompassing Router (SPIEDER)

Summary

- Hash-based traceback is viable
  - With reasonable memory constraints
  - Supports common packet transforms
  - Timely tracing of *individual* packets
- Publicly available implementations
  - FreeBSD/Linux versions available now
  - SPIEDER-based solution in development

http://www.ir.bbn.com/projects/SPIE