Efficient Network-wide Flow Record Generation

Joel Sommers
Colgate University
jsommers@colgate.edu

Rhys Bowden
University of Adelaide
rhys.bowden@adelaide.edu.au

Brian Eriksson
University of Wisconsin
beriksson@wisc.edu

Paul Barford
University of Wisconsin
pb@cs.wisc.edu

Matthew Roughan
University of Adelaide
matthew.roughan@adelaide.edu.au

Nick Duffield
AT&T Labs-Research
duffield@research.att.com
Motivation

• Need to evaluate a new protocol, system, or algorithm

• Test under a range of realistic conditions
  • Subject to Internet-like conditions
  • Use empirical data to conduct a trace-based analysis

• Need control and repeatability

• Need to measure characteristics of interest
Internet trace data

- Flow: a conversation between network entities, as observed at a given vantage point [Claffy et al., 1994]
- Unidirectional sequence of packets sharing IP source & destination addresses, protocol, source & destination ports
- Flow data commonly collected in service provider infrastructures
- Standard formats exist and are implemented widely
  - Cisco Netflow, IPFIX [RFC 5101]
Trace collection environments

- Live Internet
  - No repeatability, no control over traffic and topology
  - No knowledge of ground truth (e.g., routing changes or anomalies)
  - Privacy concerns
- Laboratory emulation (WAIL, Emulab)
  - Hard to carry out very large scale experiments
- Simulation (ns-2, ns-3, SSFnet, etc.)
  - Current simulators still don’t scale well to very large topologies and traffic configurations
**fs: a flow export record generator**

- Scale to large network and traffic configurations
  - Key: flowlets, a higher-level abstraction than packets
- Generated data should be representative of the live Internet
  - Key: leverage work from an existing network traffic generator
  - Include ability to create types of anomalous traffic flows
- Quickly synthesize new data sets
  - Produce data in widely-used formats, e.g., Netflow
fs configuration

- Declarative scenario description
  - Graphviz DOT file format [Bilgin et al., 2010]

Scenario contains specifications for:

- Topology
  - Nodes and links, including link reliability

- Traffic
  - Benign and anomalous
Traffic configuration

- **Traffic modulators** manage traffic generators
  - Modulator controls a specified number of generators over time
  - Can specify three separate modulator phases: emerge, sustain, withdraw

- Two types of generators
  - Harpoon TCP flows
    - Challenge 1: Harpoon relies on TCP implementations on end hosts
    - Challenge 2: the packet abstraction does not exist in fs
  - Simple constant or variable rate flows
    - Emulate simple UDP/TCP/ICMP flows and some types of traffic anomalies
Harpoon overview

• Scalable generation of TCP and UDP traffic flows
• Reproduce important features of measured traffic
  • Match temporal (byte, packet, flow) volumes, including diurnal characteristics
  • Relative frequency of IP sources and destinations
• Self-configures from measured flow records

Traffic generation in $fs$

- To adapt Harpoon model, leverage existing TCP modeling work to estimate flow rates and durations
  - Current implementation incorporates Mathis et al. and Cardwell et al. models
  - Models require round-trip delay and loss rate
- Simple capabilities included in $fs$ for generating simple constant- and variable-bit rate flows
  - Can also mimic flooding attacks and other types of anomalies
Key abstraction: flowlets

- Instead of packets as the core abstraction, fs uses flowlets
  - Flowlet is the main entity scheduled by the fs simulator core
  - Higher-level abstraction significantly reduces number of scheduler events

- At discrete intervals, compute how much of a flow (bytes/packets) should be emitted
  - Include relevant TCP flags in each flowlet
  - At routers/nodes, flowlets are aggregated and stored
Implementation

- At core, a discrete-event simulator
- About 2,000 lines of Python (v2)
  - Several additional Python modules used for core functionality
  - Much care taken to avoid creating new memory objects
- Bits of configuration are directly executed by the interpreter
Experiments

• Validation
  • How do the generated data compare with ns-2?
  • How do the generated data compare with data collected from a laboratory environment using the “real” Harpoon?

• Performance
  • How fast is it? How much memory does it use?

• Application domain illustration
  • How might $f_s$ be used in anomaly detection experiments?
ns-2 comparison

Nearly all fs byte and packet volumes fall within 95% C.I. for ns-2 volumes
Sensitivity to TCP model

- No packet loss in the ns-2 simulation
- Non-zero loss rates needed to make good match for each TCP throughput model
- For Mathis et al. (MSMO97) model, poor match for small flows (as expected)
- For Cardwell et al. (CSA00) model, much better match
- Results show inadequacy of current TCP throughput models
Impact of network congestion

- If aggregate flowlet volume exceeds configured link capacities, flowlets are queued.
  - Coarse approximation of packet queueing.
- Results diverge for 1 second aggregation, but are much closer for 30 second aggregation.
## Performance

<table>
<thead>
<tr>
<th>new flows per sec</th>
<th>Runtime (sec)</th>
<th>Peak memory (kB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ns-2</td>
<td>$fs$</td>
</tr>
<tr>
<td>100</td>
<td>1358</td>
<td>149</td>
</tr>
<tr>
<td>200</td>
<td>4440</td>
<td>297</td>
</tr>
<tr>
<td>400</td>
<td>15894</td>
<td>597</td>
</tr>
<tr>
<td>800</td>
<td>59661</td>
<td>1185</td>
</tr>
</tbody>
</table>
Anomaly detection experiment

• Compare two simple anomaly detection methods

  • Thresholded differencer and thresholded high-pass filter

• Superficially, the differencer performs best

  • Differencer detects every anomaly through the 30th trial

• But we cannot say anything statistically rigorous until we’ve run about 800 trials!
Summary and future work

• Ability to quickly synthesize network measurements is critical for a wide variety of networking experiments

• $fs$ quickly generates flow export data consistent with measurements collected in live networks

• Code available: http://bitbucket.org/jsommers/fs

• Future work

  • Investigate alternative models for estimating TCP flow throughputs
  
  • More flexible routing framework
  
  • How can a traffic matrix be used to generate an $fs$ configuration?
Additional slides
Examples networking problems using flow export data

- Traffic matrix estimation [Vardi, 1996; Zhang et al., 2003]
  - Estimate traffic demands between source and destination nodes using link load data
  - Flow export data used to improve and validate estimates

- Network-wide traffic anomaly detection [Barford et al., 2002, Lakhina et al., 2004; Eriksson et al., 2010]
  - Using network-wide flow or link load measurements, identify anomalous traffic conditions
## Environments for network experiments

<table>
<thead>
<tr>
<th></th>
<th>live internet</th>
<th>laboratory</th>
<th>simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>repeatability</strong></td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td><strong>supports large scale</strong></td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td><strong>ease of use</strong></td>
<td>low</td>
<td>low</td>
<td>medium-high</td>
</tr>
<tr>
<td><strong>control</strong></td>
<td>low</td>
<td>medium-high</td>
<td>high</td>
</tr>
<tr>
<td><strong>realism</strong></td>
<td>yes</td>
<td>difficult</td>
<td>difficult</td>
</tr>
</tbody>
</table>
Related work

“Models [and tools] should be specific to the research questions being investigated.” [Internet Research Needs Better Models, Floyd & Kohler, 2004]

- Need for simulation in anomaly detection studies [Ringberg et al., 2008]
  - Lack of ground truth and privacy concerns make live Internet traces problematic

- FLAME tool [Brauckhoff et al., 2008]
  - Limited modification of existing flow records to introduce anomalies

- Traffic generation
  - Harpoon [Sommers and Barford, 2004]

- Network simulators
  - ns-2 [McCanne et al., 1997], ns-3 [Henderson, et al., 2011], SSFnet [Cowie et al., 1999]
**Harpoon model**

**session level**
host pairs engaging in file transfers

**connection level**
file transfers between a given host pair using host’s TCP stack

The number of active sessions is modulated to achieve desired volumes.

Source and destination addresses are assigned to active sessions to obtain desired spatial distribution.

Inter-connection times (connection schedule)
Harpoon: reproducing traffic volumes

- **Byte volumes**
  - Measured trace vs Harpoon
  - CDF graphs showing data distribution
- **Packet volumes**
  - Measured trace vs Harpoon and harpoon - scaled
  - CDF graphs showing data distribution
- **Flow volumes**
  - Measured trace vs Harpoon
  - CDF graphs showing data distribution
Example topology

A

weight: 10
delay: 43 ms

B

10.2.0.0/16

C

weight: 30
delay: 123 ms

10.3.0.0/16
10.0.0.0/8

10.1.0.0/16
10.128.0.0/9
Topology configuration

graph threenodenetwork {
  // three node declarations, each with ipdest attributes
  a [
    ipdests="10.1.0.0/16 10.28.0.0/9"
  ];

  b [
    ipdests="10.2.0.0/16"
  ];

  c [
    ipdests="10.3.0.0/16 10.0.0.0/8"
  ];

  // three link declarations, each with weight, capacity, and delay attributes
  a -- b [ weight=10 capacity=1Gbps delay=43ms ];
  b -- c [ weight=10 capacity=1Gbps delay=31ms ];
  a -- c [ weight=30 capacity=1Gbps delay=123ms ];
}
**fs measurement data**

- **SNMP**
  - Simple interface byte/packet counts
  
  ```
  1278287633.236 a->b 10479531 bytes 9359 pkts 968 flows
  1278287643.236 a->b 10628513 bytes 9587 pkts 1023 flows
  1278287653.236 a->b 10692324 bytes 9539 pkts 983 flows
  ```

- **Flow export records**
  - Different sampling regimes supported
  - Basic text format
  - Netflow v5
  - Partial IPFIX support

```textexport a 1296521577.999705 1296521577.999705 1296521578.397185 
10.1.255.206:443->10.3.1.171:5329 tcp 0x0 harpoon 7 6535 FSA```
Topography anomalies

- **Two ways to introduce link failures**
  - Specific failure and repair times
  - Time to failure and time to repair distributions
  - Routes are instantaneously recomputed upon failure

```
graph unreliablelinks {
  a [ ipdests="192.168.42.1/24" ];
  b [ ipdests="10.1.0.0/16" ];
  c [ ipdests="10.1.42.0/24" ];
  // method 1: specific down/up times for links
  a -- b [
    weight=5, capacity=1000000000, delay=0.020,
    reliability="failureafter=30 downfor=10"
  ];
  // method 2: statistical profile of link reliability
  b -- c [
    weight=13, capacity=1Gbps, delay=10ms,
    reliability="mttf=exponential(1.0/600.0) 
                 mttr=exponential(1.0/5.0)"
  ];
}
```
Traffic configuration

Traffic flow from A to C

A

10.1.0.0/16

10.128.0.0/9

B

10.2.0.0/16

C

10.3.0.0/16

10.0.0.0/8

a [
    autoack="True" ipdests="10.1.0.0/16 10.128.0.0/9"
    traffic="m1"
    // build up and withdrawal of source s1:
    // 10 sources for a duration of 60 sec,
    // followed by 20 sources for 60 sec, ...
    m1="modulator start=0.0 generator=s1
        profile=((60,), (10, 20, 30, 20, 10))"

    // a basic Harpoon traffic setup
    s1="harpoon ipsrc=10.1.0.0/16 ipdst=10.3.0.0/24
        flowsize=pareto(10000,1.2)
        flowstart=exponential(100.0)
        sport=randomchoice(22,80,443)
        dport=randomunifint(1025,65535) mss=1460
        lossrate=randomchoice(0.001) tcpmodel=msmo97"
  ];
Simple traffic flows and anomalies

- **rawflow** generator used for defining simple constant- and variable-rate flows

- Couple with a modulator to create a SYN flood with ramp-up, sustained attack, and step-down

- **subtractive** anomaly “generator” can randomly remove flow records along a path

- Mimic data collection outages and other operational problems

```plaintext
// UDP variable-bit rate flow
s2="rawflow ipsrc=10.1.1.5/32
    ipdst=10.3.2.5/32
    flowlets=100 ipproto=udp dport=4444
    sport=randomunifint(1024,65535)
    pkts=normal(10,1) bytes=normal(1000,100)
    interval=1.0"

// SYN flood
m4="modulator start=10 generator=syns
    emerge=((1,),frange(0,100,10))
    sustain=((30,), (100,))
    withdraw=((1,),frange(100,0,-10))"
syns="rawflow ipsrc=10.1.0.0/16
    ipdst=10.4.5.0/26 dport=80
    sport=randomunifint(1,65535)
    ipproto=tcp pkts=1 bytes=40 tcpflags=SYN
    flowlets=1 interval=exponential(1/1.0)"

// Random subtractor
sub1="subtractive dstnode=b
      action=removeuniform(0.1)"
```
Results summary

- *fs* packet/byte/flow volumes match ns-2 well over intervals of 1 second or longer
- *fs* volumes also match “real” Harpoon volumes
- Variability of byte/packet volumes suggests *fs* results are statistically identical to ns-2 and Harpoon over a range of time scales (1 second or greater)
  - Poor match over intervals shorter than 1 second
- Our results expose deficiencies of TCP throughput models
- *fs* produces results much faster than ns-2, and with significantly lower memory overhead
Time-aggregation effects

- *fs* uses a fixed discrete interval for generating flowlets
- Results show expected poor match for intervals below 1 second (1 second is default interval)
Nearly all fs byte and packet volumes fall within 95% C.I. for Harpoon volumes.
Comparison of traffic variability

- Compare variability of byte counts over a range of time scales
- Results suggest that $fs$ measurements at aggregations of 1 second and greater are statistically indistinguishable from ns-2