CS 591B1: ALGORITHMIC ASPECTS OF COMPUTER NETWORKS **Spring 2002**

Lecture 2 — January 28, 2002

Lecturer: M. Mitzenmacher, J. Byers *BOSTON UNIVERSITY* *Scribe: Trevor MacDowell*

In today's lecture Prof. Michael Mitzenmacher presented his method of compressing **Bloom Filters** so that they may be efficiently transmitted as messages (minimizing either transmission size or false-positive rate). We then discussed the possibility of applying Bloom Filters to the problem of IP Traceback as motivated by the paper *Hash-Based IP Traceback* [5]. Professor John Byers concluded with an Information Theoretic basis for the concept of **Entropy** and **Shannon's Theorem** [4].

2.1 Compressed Bloom Filters

Motivation: Many applications of Bloom Filters (BFs), such as web caching, require that they be transmitted as messages. Since data transmission is, in general, more expensive than data storage it would be useful to find a method of compressing BFs . Indeed, we will see that compression in Bloom Filters comes at the cost of much higher local storage.

Recall the 3 BF tradeoff variables:

- m/n (number of bits per entry)
- k (number of hash functions)
- f (false-positive rate)

In the previous lecture we learned that with good hash functions $\Pr[\text{a given bit of the } BF = 1] = \frac{1}{2}$. So it does not seem, at first glance, that compression is possible. But allowing m/n to grow while keeping the compressed transmission size per entry (z/n) constant will allow us to further lower the false-positive rate (i.e. we will create a sparser filter so that compression is feasible). Similarly we could keep the false-positive rate constant and optimize for transmission size.

Given z and n choose m and k to optimize[1] f:

1. $p \approx e^{-\frac{kn}{m}}$
2. $f \approx \left(1 - e^{-\frac{kn}{m}}\right)^k$
3. $z = m\mathcal{H}(p)$

Optimization through calculus[2]: $k = \frac{m(\ln(2))}{n}$

Note that the optimal BF without compression yields the worst results when compression is permitted. So surprisingly we can always improve performance through compression.

[1] Here we will assume the existence of an optimal compressor, $z = m\mathcal{H}(p)$ where $\mathcal{H}(\cdot)$ is the entropy function.

[2] Please see [2] for proof of this, and a graph showing optimization points of BF vs. CBF.

2.2 Hash-Based IP Traceback

The goal of Hash-Based IP Traceback is to be able to "identify the source of any piece of data sent across the network."[5] Specifically we are thinking of the attacker-victim model where a victim (a host or network that is being, or has just been, attacked) would like to determine the source of an attack. The Source Path Isolation Engine (SPIE) system will enable a victim to do this as long as the attack is reported quickly, and was sent through SPIE enabled routers[3]. The system stores digests of packets that cross the network in bloom filters. Given a request from a victim, the system uses these digests to build an attack graph by first determining the ingress points of 'attack packets' on its own network, and then forwarding the request to SPIE router s along those paths.

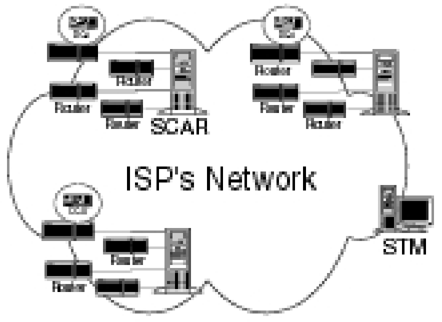

Table 2.1. SPIE Inrastructure [5]

Three major components of SPIE infrastructure:

1. DGAs (Data Generation Agents): Each router on an SPIE enabled network has a DGA that produces packet digests (containing information about where each packet came from, and enough of the header/payload to uniquely identify it). These digests are then efficiently stored in bloom filters at the DGA for a short amount of time. If a particular time period needs to be examined further then the data is transmitted to the SCAR for analysis and longer term storage.

2. SCARs (SPIE Collection and Reduction Agents): Each SCAR is responsible for traceback within a given region of the network. Upon request it will gather information about a certain time period from each of the DGAs in its region and produce a local attack graph.

3. STM (SPIE Traceback Manager): Each network contains one STM which is responsible for controlling the SPIE system. It receives and processes requests from victims, propagates those requests to the SCARs, and pieces together the graphs generated by each SCAR to produce a complete attac k graph.

[3]SPIE may be able to narrow down an attack region if most of the routers traversed were SPIE enabled.

2.3 Entropy and The Entropy Function

For our purposes, entropy is a measure of uncertainty as it relates to the information gained by conducting experiments from a given sample set. Through the realization of Shannon's Theorem we will also see that entropy can tell us the minimum number of bits required (ideally) to represent a string. Before proceeding we need to define an *experiment* in this context.

2.3.1 Experiments

An experiment is a set of possible outcomes where each outcome is represented by the probability of its occurrence. These probabilities will sum to 1 and must cover the entire event space.

- Experiment=$(p_1, p_2, \ldots, p_n)$ where $p_i = \Pr[\text{event } e_i \text{occurs}]$
- $\sum_{i=1}^{n} p_i = 1$
- In general, as n grows large, so does the level of uncertainty about an experiment's outcome.

Example: $\left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$ is an experiment with n possible outcomes each of which are equally likely.

2.3.2 Requirements for a definition of entropy

We will attempt to define a funtion $\mathcal{H}(\cdot)$ that, given a sample space, will output a measure of the uncertainty associated with an experiment in that space. Any definition of uncertainty (entropy) must follow *all* of the following criteria:

1. $\mathcal{H}\left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right) < \mathcal{H}\left(\frac{1}{n+1}, \frac{1}{n+1}, \ldots, \frac{1}{n+1}\right) \forall n \in \mathbb{Z}^+$ i.e. we expect uncertainty to grow as the sample size gets larger.
2. $\mathcal{H}(p_1, p_2, \ldots, p_n)$ is defined and continuous for all $(p_1, p_2, \ldots, p_n)$ satisfying $0 \le p_i \le 1$, $\sum_{i=1}^{n} p_i = 1$.
3. $\mathcal{H}\left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right) = \mathcal{H}\left(\frac{b_1}{n}, \frac{b_2}{n}, \ldots, \frac{b_k}{n}\right) + \sum_{i=1}^{k} \frac{b_i}{n} \cdot \mathcal{H}\left(\frac{1}{b_i}, \frac{1}{b_i}, \ldots, \frac{1}{b_i}\right)$ for arbitrary positive integers b_i satisfying $\sum_1^k b_i = n$ (blocks).

The third requirement simply means that if we break the sample space into k blocks that the uncertainty associated with 'choosing' an element from the original space should be the same as the uncertainty associated with first 'choosing' a block, and then an element within that block:

$$\underbrace{\mathcal{H}\left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)}_{\textit{One pass info. gain}} = \underbrace{\mathcal{H}\left(\frac{b_1}{n}, \frac{b_2}{n}, \ldots, \frac{b_k}{n}\right)}_{\textit{1st pass (choose a block)}} + \underbrace{\sum_{i=1}^{k} \frac{b_i}{n} \cdot \mathcal{H}\left(\frac{1}{b_i}, \frac{1}{b_i}, \ldots, \frac{1}{b_i}\right)}_{\textit{2nd pass (choose element within block)}}$$

Aside: Given a sample space where all events are not equally likely we can always use this property to transform the space into equiprobable blocks.

2.3.3 The Entropy Function

Only one family of functions satisfies all three of these criterion[3]:

$$\boxed{\mathcal{H}_l(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{k} p_i \cdot \log_l(p_i) = \sum_{i=1}^{k} p_i \cdot \log_l\left(\frac{1}{p_i}\right)}$$

The base l allows for the scaling of entropy (although in practice we will almost exclusively work with $l = 2$).

<u>Some Examples:</u>

1. $\mathcal{H}\left(\frac{1}{2},\frac{1}{2}\right) = \frac{1}{2}\cdot\log_2(2) + \frac{1}{2}\cdot\log_2(2) = 1$
2. $\mathcal{H}\left(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4}\right) = 4\cdot\left(\frac{1}{4}\cdot\log_2(4)\right) = 2$
3. $\mathcal{H}\left(\frac{1}{4},\frac{3}{4}\right) = \frac{1}{4}\cdot\log_2(4) + \frac{3}{4}\cdot\log_2\left(\frac{4}{3}\right) \approx 0.8113$

2.3.4 Shannon's Noiseless Coding Theorem [4]

Prefix-free Encoding

A prefix-free encoding of sample space is a set of codes $\langle c_1,\ldots,c_n\rangle$ such that:

- c_i is a bit seqence representing event i, and p_i is the probability that event i occurs.
- $\forall i,j \in \mathbb{Z}^+$: c_i is not a prefix of c_j.
- length(c_i) =the number of bits in c_i

Noiseless Coding Theorem

For any prefix-free encoding $\langle c_1,\ldots,c_n\rangle$ (in bits) of a sample space $(p_1,p_2,\ldots,p_n)$:

1. $\mathcal{H}(p_1,p_2,\ldots,p_n) \le \underbrace{\sum_{i=1}^{n} p_i\cdot\text{length}(c_i)}_{\textit{Expected bits per symbol}}$
2. $\exists$ codes $\langle c_1,\ldots,c_n\rangle$ s.t. $\sum_{i=1}^{n} p_i\cdot\text{length}(c_i) \le \mathcal{H}(p_1,p_2,\ldots,p_n)+1)$

In 1952 Huffman was able to produce simple codes that always meet the bounds set in Shannon's Theorem. Shannon's second bound has since been further improved, and it is now known that codes exist that acheive encoding within ϵ of the sample set's entropy (where $\epsilon > 0$).

Example

Consider the following sample space and its Huffman coding:

p_i	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$
c_i	0	10	110	111

$\sum_{i=1}^{n} p_i\cdot\text{length}(c_i) = \frac{1}{2}\cdot\text{length}(c_1)+\frac{1}{4}\cdot\text{length}(c_2)+\frac{1}{8}\cdot\text{length}(c_3)+\frac{1}{8}\cdot\text{length}(c_4) = \frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2+\frac{1}{8}\cdot 3+\frac{1}{8}\cdot 3 = 1\frac{3}{4}$

$\mathcal{H}\left(\frac{1}{2},\frac{1}{4},\frac{1}{8},\frac{1}{8}\right) = \frac{1}{2}\cdot\log_2(2)+\frac{1}{4}\cdot\log_2(4)+\frac{1}{8}\cdot\log_2(8)+\frac{1}{8}\cdot\log_2(8) = 1\frac{3}{4}$

Bibliography

[1] B. Bloom. "*Space/time trade-offs in hash coding with allowable errors*," Communications of the ACM, 13(7):422-426, 1970.

[2] M. Mitzenmacher. "*Compressed Bloom Filters*," in Proceedings of PODC 2001.

[3] S. Roman. "*Coding and Information Theory*," Springer, New York, 1992.

[4] C. E. Shannon. "*A Mathematical Theory of Communication,*" Bell System Tech Journal 27, p379-423 and p625-656. 1948.

[5] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, S. Kent, and T. Strayer, "*Hash-Based IP Traceback*," in Proceedings of ACM SIGCOMM '01.