(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 December 2006 (07.12.2006)

PCT

(10) International Publication Number
WO 2006/130840 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2006/021512

(22) International Filing Date: 2 June 2006 (02.06.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/686,560	2 June 2005 (02.06.2005)	US
60/686,570	2 June 2005 (02.06.2005)	US
60/689,651	10 June 2005 (10.06.2005)	US
60/709,191	17 August 2005 (17.08.2005)	US
60/709,198	17 August 2005 (17.08.2005)	US

(71) Applicant *(for all designated States except US)*: **GEORGIA TECH RESEARCH CORPORATION** [US/US]; 505 Tenth Street, Atlanta, GA 30332-0415 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **XU, Jun** [CN/US]; 375 Highland Avenue, #203, Atlanta, GA 30312 (US). **ZEGURA, Ellen, W.** [US/US]; 2458 Pangborn Circle, Decatur, GA 30033 (US). **KUMAR, Abhishek** [IN/US]; 914 Collier Road, Apt. 3 3218, Atlanta, GA 30318 (US). **SUNG, Minho** [KR/US]; 252 Devonshire Drive, Alpharetta, GA 30022 (US).

(74) Agents: **NORTON, Lisa, K.** et al.; DLA PIPER RUDNICK GRAY CARY US LLP, P.O. Box 9271, Reston, VA 20195 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 2006/130840 A2

(54) Title: SYSTEM AND METHOD FOR DATA STREAMING

(57) Abstract: A system and method for estimating flow size distribution and subpopulation flow size distribution.

TITLE

SYSTEM AND METHOD FOR DATA STREAMING

Jun Xu
Ellen Zegura
Abhishek Kumar
Minho Sung

This application claims priority to US Provisional Patent Application 60/689,651 entitled "Data Streaming Algorithms for Efficient Estimation of Flow Size Distribution," filed June 10, 2005; US Provisional Patent Application 60/686,560 entitled "Data Streaming Algorithm for Estimating Subpopulation Flow Size Distribution," and filed June 2, 2005; US Provisional Patent Application 60/686,570 entitled "Data Streaming Algorithms for Accurate and Efficient Measurement of Traffic and Flow Matrices," filed June 2, 2005; US Provisional Patent Application 60/709,198 entitled "Data Streaming Algorithms for Detection of Super Sources and Super Destination," filed August 17, 2005; and US Provisional Patent Application 60/709,191 entitled "Including Network Data Streaming in a Broad Architecture for Network Monitoring Applications," filed August 17, 2005; all of which are incorporated herein by reference. This application is also related to the PCT Patent Application filed on June 2, 2006 entitled "System and Method for Measuring Traffic and Flow Matrices" which is also incorporated herein by reference.

This invention was made with Government Support under Grant No. ANI-0238315 awarded by the National Science Foundation of the United States. The Government has certain rights in the invention.

BRIEF DESCRIPTION OF THE FIGURES

FIGURE 1 illustrates an example of a network monitoring device, according to one embodiment.

FIGURE 2 illustrates another example of a network monitoring device, according to one embodiment.

FIGURE 3 illustrates an example of a network that includes the network monitoring device in accordance with the invention, according to one embodiment.

FIGURES 4A and 4B illustrate an example of a counter array sketch data structure used with the network monitoring device, according to one embodiment.

FIGURES 5A and 5B illustrate an example of a bit map sketch data structure used with the network monitoring device, according to one embodiment.

FIGURE 6 illustrates a method of estimating flow size distribution, according to one embodiment.

FIGURE 7 illustrates another embodiment used to estimate the storage needs of the sketch, referred to as a Multi-Resolution Array of Counters (MRAC) scheme.

FIGURE 8 illustrates pseudocode that illustrates a method for mapping a virtual array to an actual array, according to one embodiment.

FIGURE 9 illustrates pseudocode that illustrates an EM algorithm for estimating ϕ_i, which is the fraction of flows that are of size i ($i = 1, 2, \ldots, z$), according to one embodiment.

FIGURE 10 illustrates an example of a network monitoring device that monitors a stream of data packets over a link of a communications network, according to one embodiment.

FIGURE 11 illustrates an overview of a method of estimating flow size distribution using an online streaming module and a sampling module, according to one embodiment.

FIGURE 12 illustrates example output of the preprocessing module 98, according to one embodiment.

FIGURE 13 is pseudocode that illustrates a method for estimating the subpopulation flow size distribution (SubFSD) from observed data, according to one embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

Monitoring the aggregate IP traffic passing through a high-speed link is a complicated task. Flows, which are groupings of packets, provide a convenient abstraction, allowing aggregation of the packet-level information into coarser-granularity flow-level information. Flow statistics, such as the total number of flows and the distribution of flow sizes, capture significant properties of the underlying network traffic, and are of interest to both service providers and client network operators.

FIGURE 1 illustrates an example of a network monitoring device 20 that monitors a stream of data packets 22 over a link (not shown) of a communications network. The device may include a collection unit 24 and an analysis unit 26. The collection unit 24 collects sampled data packets or generates a sketch of the data packet stream over the link during a period of time while the analysis unit 26 analyzes the sampled data packets or the sketches. The collection unit and the analysis unit may be implemented in software, but may also be implemented in hardware or in a combination of hardware and software. The collection unit and analysis unit may be co-located or may be located at different physical locations and may be executed on the same piece of hardware or different pieces of hardware. The sketch is a data structure that stores information about each data packet in a packet stream wherein the sketch is typically a smaller size than the actual data packets. Two examples of a sketch that can be used with the network monitoring device are shown in FIGURES 4A-4B and 5A-5B, respectively.

The collection unit 24 may further comprise one or more of a selection process 28, an online streaming module 32 and a reporting process 30 which are each a piece of software code that implements the functions and methods described below. The selection process performs a sampling of the packet stream and selects the sampled data packets that are communicated to the reporting process 30 that aggregates the sampled data packets and generates a report based on the sampled data packets. The online streaming module 32 monitors the packet stream and generates one or more sketches (shown in FIGURES 4A, 4B

and/or 5A, 5B) based on the data packets in the packet stream. As shown, the collection unit may communicate flow records/reports and the sketches to the analysis unit 26.

The analysis unit 26 may further comprise a collector 34, a digest collector 36 and one or more network monitoring applications 38, such as application 38_1, application 38_2 and application 38_3. The collector 34 receives the flow records/reports from the reporting process 30 while the digest collector 36 receives the sketches generated by the online streaming module 32. The flow records/reports and/or the sketches may then be input into the monitoring applications 38 that perform different network monitoring functions. For example, one application can generate a data packet volume estimate over a link between a first and second node of a network, another application may generate a flow estimate between a flow source and a flow destination. In general, the network monitoring device 20 is a platform on which a plurality of different monitoring applications can be executed to perform various network monitoring functions and operations. One or more examples of the monitoring applications that may be executed by the network monitoring device are described in more detail below.

FIGURE 2 illustrates another example of a network monitoring device 20 that includes the collection unit 24 and the analysis unit 26. In this example, the collection unit 24 includes only the online streaming module 32 that generates the sketches and periodically communicates the sketches to the analysis unit 26 that includes the offline processing module 38. Thus, in this example, the network monitoring device 20 generates sketches and then analyzes those sketches to perform a network monitoring function. Each data packet 22_1, 22_2, 22_3 and 22_4 may include a header 23_1, 23_2, 23_3 and 23_4 that may be used by the online streaming module 32 to generate the sketches as described in more detail below. In this example, a user may submit a query, such as the traffic volume of a particular link, to the monitoring application and the monitoring application returns a result to the user, such as the traffic volume of the particular link based on the sketches generated by the online streaming module 32.

FIGURE 3 illustrates an example of a network 40 that includes the network monitoring device in accordance with the invention. The network may include one or more nodes 42, such as routers 42_1, 42_2, 42_3, 42_4 and 42_5, wherein the network may include one or more ingress routers and one or more egress routers. The routers form the network that permits data packets to be communicated across the network. The links between the nodes

over which data packets are communicated are shown by solid lines. In this example of the network, the collection unit 24 is physically located at each link interface at each node 42 and is a piece of software executed by each router. Furthermore, the analysis unit 26 may be physically located at a central monitoring unit 44, such as a server computer, that is remote from the nodes of the network and the collection unit is a piece of software executed by the central monitoring unit 44. Each collection unit 24 may generate one or more sketches for its link during a particular time period and then communicate those sketches to the central monitoring unit as shown by the dotted lines in FIGURE 3.

FIGURES 4A and 4B illustrate an example of a counter array sketch data structure 50 used with the network monitoring device. The sketch data structure may include one or more counters 52 (C1 to Cb for example) in an array (known as a counter array) wherein each counter has an index number (1 to b in the example shown in FIGURE 4A) associated with the counter. Each counter can be incremented based on the scanning of the data packets in the data stream performed by the online streaming module 32 shown above. Furthermore, each counter is associated with a particular set of one or more packet flow label attributes. Each data packet flow label (typically contained in the header portion of the data packet and having 13 bytes) may include a field containing a source node address (an address of the source of the particular data packet, a field containing a destination node address (an address of the destination of the particular data packet, a field containing a source port (an application at the source from which the particular data packet is generated), a field containing a destination port (the application at the destination to which the particular data packet is being sent) and a field containing a protocol designation that identifies the type of protocol being used for the particular data packet, such as HTTP, UDP, SNMP, etc. For example, one counter for a particular network may be assigned to count all data packets (during a predetermined time interval) that are sent from a particular source node while another counter may be assigned to count all data packets (during a predetermined time interval) that are sent to a particular application in a particular destination node. Thus, the assignment of each counter in the counter array is configurable depending on the particular network and the particular user and what the particular user needs to monitor in the network.

FIGURE 4B illustrates an example of a piece of pseudocode 54 that implements the counter array data structure shown in FIGURE 4A. The pseudocode shows that, during an initialization process 56 (which may occur when the monitoring is started or to reset the

sketch data structure when the predetermined time period (an epoch period such as 5 minutes) has expired), each counter in the counter array is reset to a default value that can be zero. During an update process 58, upon the arrival on each data packet, a hash function is performed on the flow label of the data packet (illustrated as h(pkt.flow_label) in the pseudocode) which generates an index value (ind) into the counter array and the counter at that index location is incremented by one to indicate that a data packet with the particular set of one or more packet flow label attributes was monitored by the particular online streaming module 32. For each data packet, only one hash operation, one memory read and one memory write (to the same location) is performed so that the counter array is able to operate at OC-768 (40 Gbps) speed with off-the-shelf *10ns* SRAM and an efficient hardware implementation of the H_3 family of hash functions.

The hash function used by the counter array (or the bitmap sketch described below) may, in one embodiment, be the known H_3 family of hash functions that are described in an article by J. Carter and M. Wegman entitled "Universal classes of hash functions", *Journal of Computer and System Sciences,* pages 143-154 (1979) which is incorporated herein by reference. Each hash function in the H_3 class is a linear transformation $B^T = QA^T$ that maps a w-bit binary string $A = a_1a_2\ldots a_w$ to an r-bit binary string $B = b_1b_2\ldots b_r$ as:

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_r \end{pmatrix} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1w} \\ q_{21} & q_{22} & \cdots & q_{2w} \\ \cdots & \cdots & \cdots & \cdots \\ q_{r1} & q_{r2} & \cdots & q_{rw} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_w \end{pmatrix}$$

Here a k-bit string is treated as a k-dimensional vector over finite field GF(2) = {0,1} and T stands for transposition. Q is a r x w matrix defined over GF(2) and its value is fixed for each hash function in H_3. The multiplication and addition in GF(2) is boolean AND (denoted as o) and XOR (denoted as $\oplus$), respectively. Each bit of B is calculated as:

$$b_i = (a_1 \circ q_{i1}) \oplus (a_2 \circ q_{i2}) \oplus \cdots \oplus (a_w \circ q_{iw}) \quad i = 1,2,\ldots,r$$

Since computation of each output bit goes through $\log_2 w$ stages of Boolean circuitry, and all output bits can be computed in parallel, the hash function can finish well within 10ns. Now, an example of another sketch data structure is described.

FIGURES 5A and 5B illustrate an example of a bitmap sketch data structure 60 used with the network monitoring device. In this example, the sketch data structure may include one or more bit positions 62 (1 to b for example) in an array (known as a bit map sketch) wherein each bit position has an index number (1 to b in the example shown in FIGURE 5A) associated with the bit position. Each bit position can have a value of "0" or "1" and be set to "1" based on the scanning of the data packets in the data stream performed by the online streaming module 32 shown above. Furthermore, each bit position is associated with a particular data packet characteristic that uniquely identifies the data packet wherein that portion of the data packet is input to the hash function. In particular, the invariant portion of a packet used as the input to the hash function must uniquely represent the packet and by definition should remain the same when it travels from one router to another. At the same time, it is desirable to make its size reasonably small to allow for fast hash processing. Therefore, the invariant portion of a packet consists of the packet header, where the variant fields (e.g., TTL, ToS, and checksum) are marked as 0's, and the first 8 bytes of the payload if there is any. As is known, these 28 bytes are sufficient to differentiate almost all non-identical packets.

FIGURE 5B illustrates an example of a piece of pseudocode 64 that implements the bit map data structure shown in FIGURE 5A. The pseudocode shows that, during an initialization process 66 (which may occur when the monitoring is started or to reset the sketch data structure when the predetermined time period (an epoch period such as 5 minutes) has expired), each bit in the bit map is reset to a default value that is, in one embodiment, zero. Then, during an update process 68, upon the arrival of each data packet (pkt), a hash function is performed on the invariant portion of the packet (ϕ(pkt) in the pseudocode) which generates an index value (ind) into the bit map and the bit at that index location is set to "1" to indicate that a data packet with the particular invariant portion has been monitored by the collection unit. The hash function used by the bitmap sketch may be the same hash function used for the counter array sketch. As with the counter array set forth above, the bit map, once it reaches a threshold level of fullness, is stored in a memory or disk and then communicated to the analysis unit as described above. As with the counter array, the bit map has the same performance characteristics as the counter array.

For each of these sketch data structures described above, the sketch data structure trades off complete information about each data packet for a limited amount of information

about each data packet in the link. Unlike a typical system in which the data packets are sampled (and only information about the sampled data packets are stored), the sketches store some information about each data packet due to the hash function and the bitmap or counter array.

Estimating Flow Size Distribution

Flow distribution information can be useful in a number of applications in network measurement and monitoring. It may allow service providers to infer the usage pattern of their networks, such as the approximate number of users with dial-up or broadband access. Such information on usage patterns can be important for the purposes of pricing, billing, infrastructure, engineering, and resource planning. In addition, network operators may also infer the type of applications that are running over a network link without looking into the details of traffic such as how many users are using streamed music, streamed video, and voice-over IP. In the future, more network applications will be recognizable through flow distribution information.

Furthermore, flow distribution information can help locally detect the existence of an event that causes the transition of the global network dynamics from one mode to another. An example of such mode transition is a sudden increase in the number of large flows (i.e., elephants) in a link. Possible events that may cause this include link failure or route flapping. Merely looking at the total load of the link may not detect such a transition since this link could be consistently heavily used anyway.

In addition, flow distribution information may also help detect various types of Internet security attacks such as DDoS and Internet worms. In the case of DDoS attacks, if the attackers are using spoofed IP addresses, we will observe a significant increase in flows of size 1. In the case of Internet worms, we may suddenly find a large number of flows of a particular size in Internet links around the same time, if the worm is a naïve one that does not change in size. Also, the historical flow distribution information stored at various links may help us study its evolution over time.

Additionally, knowing the flow distribution of each link may help other network measurement applications, such as traffic matrix estimation. It is possible to use tomography

techniques to infer the traffic matrix from link load and aggregate input/output traffic. Flow distribution at each node will make such tomography much more accurate, since it allows the correlation of not only the total traffic volume (load), but also the correlation of its distribution into different flow.

Multiple other uses of flow distribution information can be seen by those of ordinary skill in the art.

FIGURE 6 illustrates an overview of a method of estimating flow size distribution, according to one embodiment. An estimate of how the total traffic volume splits into flows of different sizes is determined. As explained earlier with respect to FIGURES 1-3, the method is run using software set up at each link of interest in a network so that the packets crossing that link can be monitored. In 607, the storage needs of the sketch are estimated. In 610, the online streaming module 32 monitors the packet stream and generates a sketch for a data packet stream run over a link during a period of time. In 615, the sketch is sent to a central server 44. In 620, the central server 44 performs algorithms analyzing the sketch. Elements 607, 610, 615, and 620 will be described in more detail below.

With respect to 607, the storage needs of the sketch can be estimated in various ways. The sketch data structure 50 is described in detail with respect to FIGs. 4A-4B. The storage needs of the sketch determines the amount of fast memory required for implementing the array of counters. In one embodiment, the number of flows expected to be seen in the sketch's time period is used as the estimated storage need of the sketch. Those of skill in the art will be able to make this determination.

FIGURE 7 illustrates another embodiment used to estimate the storage needs of the sketch, referred to as a Multi-Resolution Array of Counters (MRAC) scheme. First, a virtual array of counters 705 that is large enough to accurately estimate the flow distribution, even in the worst case, is determined. Then, virtual array 705 is mapped or folded to actual array 710.

FIGURE 8 illustrates pseudocode that illustrates a method for mapping the virtual array 705 to the actual array 710, according to one embodiment. Note that a base-2 version ($\log_2$) is described, but that the algorithm may be generalized to any arbitrary base. Referring to FIGURE 8, in the base-2 version, the virtual array 705 of $M = 2^r m$ counters is mapped to $r+1$ actual arrays of size m each. Half of the hash space will be mapped to (folded into) array

1 (A_1); half of the remaining has space (i.e., 1/4 of the total hash space) will be mapped to array 2 (A_2), and so on. Finally, we are left with two blocks of hash space of size m each. They are directly mapped to arrays r and $(r + 1)$. The total space taken by the arrays is m $(\log_2 ((M/m) + 1))$.

The actual mapping/folding algorithm is shown in FIGURE 8. As described above, the arrays $A_1, A_2, \ldots, A_r, A_{r+1}$ cover the respective hash ranges of [0, 1/2 M), [1/2M, 3/4M), [3/4M, 7/8M), ..., $[(1 - 1/2^{r-1})M, (1 - 1/2^r)M)$, $[(1 - 1/2^r)M, M)$. If a hash index *ind* is mapped to an array, the counter indexed by (*ind* mod *m*) in that array will be incremented. Therefore, the values of 2^{r-1} counters in the virtual array map to (fold into) 1 counter in array A_1, and the values of 2^{r-2} virtual counters map to 1 counter in array A_2, and so on. The $(r + 1)$ arrays together cover the entire virtual hash space. The regions covered by any two arrays are disjoint.

Thus, Array A_1 processes approximately 1/2 of the flows (i.e., every packet in approximately half of the flows), array A_2 processes approximately 1/4 of the flows, and so on.

Referring now to 610 of FIGURE 6, and also FIGURES 1-2 and 4, the online streaming module 32 monitors the packets 22 and generates a sketch 50 for a data packet stream run over a link during a period of time. For each packet 22, the online streaming module 32 computes exactly one hash function and increments exactly one counter 52. Upon arrival of a packet 22 at the link, its flow label 23 (which can be any combination of fields in the header) is hashed to generate an index into this array and the counter at this index is incremented by 1. Collisions due to hashing might cause two or more flow labels 23 to be hashed to the same indices. Counters 52 at such an index would contain the total number of packets belonging to all of the flows colliding into this index. It should be noted that because the sketching process is very simple and fast, the online streaming module 32 can operate at very high speeds without missing any packets.

Referring now to 620 of FIGURE 6, the central analysis program runs algorithms analyzing the sketches. There are several estimations that can take place, including: the total number of flows during a sketch, the number of flows of size 1, and the number of flows greater than size 1.

Estimate the total number of flows. The total number of flows during the measurement interval of the sketch can be estimated. This is done by looking at the number of counters that are equal to 0, and the number of counters that are not equal to 0, and then combine this with an estimate of the collisions. In an array of m counters, if the number of zero entries is m_0 after the insertion of n flows, the maximum likelihood estimator for n is:

$$\tilde{n} = m \ln (m/m_0)$$

Estimate the number of flows of size 1. Note that a counter of value 1 must contain exactly one flow of size 1 (*i.e.*, there must be no collision). The total number of flows containing exactly one packet is significant. From a modeling perspective, this number helps affirm or reject statistical hypotheses regarding flow size distribution. In addition, as explained above, among other things, this number helps determine abnormal or malicious behavior, such as port-scanning and DDoS attacks, because such behavior often manifests itself as a significant increase in the number of flows of size 1.

To estimate the number of flows of size 1, the following formula is used:

$$n_1 = y_1 e^{n/m}$$

where:

n_1 = the number of flows of size 1
n = the overall number of flows
y_1 = the number of counters with a value of 1
y = the number of counters

Estimating the number of flows of a size greater than 1. For flows of size greater than 1 (i.e., 2, 3, etc.), an additional process can be used to estimate flow distribution. While a counter of value 1 cannot be involved in a collision, counter-values of 2 and above could be caused by the collision of two or more flows. For example, among the counters of value 2, some correspond to a flow of size 2, while the others could be two flows of size 1. Thus, while the estimate for n_1 (the number of flows of size 1) depends only on the estimate of n, the estimate of n_2 will depend on both n and n_1. More generally, the estimate of n_i will depend on our estimates of n, n_1, n_2, . . ., n_{i-1}. Thus, for a large flow size i, the estimate is more susceptible to errors due to this cumulative dependence effect, resulting in a sharp increase in estimation errors. Therefore, to accurately estimate flow distribution, an approach

based on Expectation Maximization (EM) method for computing Maximum Likelihood Estimation (MLE) can be utilized in one embodiment.

The MLE algorithm computes the flow distribution that is most likely to result in the observed counter values after the hash collisions. EM is used to interactively compute the local MLE. In one embodiment, EM is used in finding MLE when the observation can be viewed as incomplete data. In this context, the observed counter values can be viewed as incomplete data, and the missing part is how flows collide with each other during hashing.

The EM algorithm, which captures intuition on handling missing data, works as follows. It starts with a guess of the parameters, and then replaces missing values by their expectations given the guessed parameters, and finally estimates the parameters assuming the missing data are equal to their estimated values. This new estimate of missing values gives us a better estimate of parameters. This process is iterated multiple times until the estimated parameters converge to a set of values, (typically a local maximum as mentioned above).

FIGURE 9 illustrates pseudocode that illustrates an EM algorithm for estimating ϕ_i, which is the fraction of flows that are of size i ($i = 1, 2, \ldots, z$), according to one embodiment. Note that z is the maximum counter value observed from an array, and that the observation y, obtained form the output of the online streaming module 32, is y_i ($i = 1, 2, \ldots, z$), the number of counters that have value i. Referring to lines 1-2 of FIGURE 9, we first need a guess of the flow distribution ϕ^{ini}, and the total number of flows n^{ini}. In the algorithm, we use the distribution obtained from the raw counter values as ϕ^{ini} and the total number of non-zero counters as n^{ini}. Based on this ϕ^{ini} and n^{ini}, we can compute, for each possible way of "splitting" an observed counter value, its average number of occurrences. Then the counts n_i for flows of corresponding sizes will be credited according to this average. For example, when the value of a counter is 3, there are three possible events that result in this observation: (i) 3 = 3 (no hash collision); (ii) 3 + 1 + 2 (a flow of size 1 colliding with a flow of size 2); and (iii) 3 = 1 + 1 + 1 (three flows of size 1 hashed to the same index). Given a guess of the flow distribution, we can estimate the posterior probabilities of these three cases. Say the respective probabilities of these three events are .5, .3 and .2, and there are 1000 counters with value 8. Then we estimate that, on the average, 500, 300, and 200 counters split in the three above ways, respectively. Finally, after all observed counter values are split this way,

we get the new counts $n_1, n_2, \ldots, n_z$, and obtain n^{new} (see line 19 of FIGURE 9). We then renormalize them into a new and refined flow distribution ϕ^{new}.

The pseudocode in FIGURE 9 also accounts for the probability of collision patterns, as set forth in lines 5-15 of FIGURE 9. Note that $p(\beta \mid \phi^{old}, n, V = i)$ is determined by:

$$p(\beta \mid \phi, n, v) = \frac{p(\beta \mid \phi, n)}{\sum_{\alpha \in \Omega_v} p(\alpha \mid \phi, n)}$$

Where the *a priori* (i.e., before observing the value v) probability that event β happens, given φ and n, can be computed as follows:

$$p(\beta \mid \phi, n) = \left(\prod_{j \in I} e^{-\lambda_j}\right)\left(\prod_{j \in I} e^{-\lambda_j}\right)\left(\prod_{i=1}^{q} \frac{\lambda_{s_i}^{f_i}}{f_i!}\right) = e^{-\lambda} \prod_{i=1}^{q} \frac{\lambda_{s_i}^{f_i}}{f_i!}$$

Here, event β is defined in lines 7 to 11 in FIGURE 9, and λ_i denotes the average number of size i flows (before collision) that are hashed to an index in the array. In other words:

$$\lambda_i = \frac{n_i}{m} = \frac{n\phi_i}{m}$$

We define:

$$\lambda = \sum_{i=1}^{z} \lambda_i$$

Estimating high counter values. For large counter values, the number of possible collisions that could give rise to the observed value is immense. To reduce the complexity of enumerating all events that could give rise to a large counter value (e.g., larger than 300), we ignore the case involving the collision of 4 or more flows at the corresponding index. Since the number of counters with a value larger than 300 is quite small, and collisions involving or more flows occur with a low probability, this assumption has very little impact on the overall estimation mechanism With similar justifications we ignore events involving 5 or more collisions for counters larger than 50 but smaller than 300, and those involving 7 or more collisions for all other counters. This reduces the asymptotic computationals complexity of "splitting" a counter-value j to $((j^3)$ (for j>300). Note that we need to do this

computation only once for all counters that have a value j, and the number of unique counter-values is quite small.

Finally, since the numbers of counters with very large values (say larger than 1000) is extremely small, we can ignore splitting such counter values entirely and instead report the counter value as the size of a single flow. This will clearly lad to a slight overestimation of the size of such large flows, but since the average flow size (approx. 10) is two to three orders or magnitude smaller than these large flows, this error is miniscule in relative terms.

Estimating Subpopulation Flow Size Distribution

The Flow Size Distribution is a fundamental statistic since its accurate estimation can enable the inference of many other statistics such as the total number of flows and the average flow size. However, there are a number of applications where the flow size distribution of a particular subpopulation (SubFSD), i.e., a subset of the complete flow population, is important. Such subpopulations may be defined by a traffic subtype (e.g., web or peer-to-peer traffic), by a source or destination IP address/prefix, by protocol (e.g., TCP or UDP), or some combination thereof. For example, to investigate a sudden slowdown in DNS lookups, a network operator may specify all flows with source or destination port 53 as the subpopulation of interest and query the data collected in the preceding hour.

The ability to monitor the SubFSD for any arbitrary subpopulation provides a powerful and versatile tool for the characterization of traffic traversing a monitoring point. Since flow size distribution is a special case of SubFSD, the motivations for estimating the flow size distribution, ranging from accounting to anomaly detection, also apply to SubFSD. In addition, the capability to zoom-in' and take a closer look at any subpopulation, supports a number of applications. Two such applications are highlighted below, although those of ordinary skill in the art will see that multiple other applications are possible.

With respect to security, DDoS attacks and worm propagation often manifest themselves as anomalous deviations in the flow distribution, such as a sudden increase in the number of flows with exactly one packet. Such deviations may not be statistically significant in the aggregate traffic, but may be clearly discernible if appropriate subpopulations are investigated. For example, in a subnet containing a DNS server and a web server, the flows of size 1 due to DNS traffic might mask out a spurt in flows of size 1 due to a SYN-flooding DDoS attack against the web server. On the other hand, looking at the SubFSD of HTTP flows will immediately reveal the attack.

With respect to traffic engineering, knowledge of the SubFSD of various subpopulations within the aggregate traffic can enable fine-grained traffic engineering. For example, the impact of routing disruptions is often limited to specific subpopulations of traffic between the Origin-Destination (OD) pairs affected by the disruption. When only statistics about the complete flow population is available, monitoring this impact may prove to be a fairly involved problem. On the other hand, with the ability to monitor arbitrary subpopulations, this impact can be easily monitored by estimating the SubFSD for the subpopulations affected by the disruption. Note again that in this example, the subpopulation of interest is not known till after the event of interest has occurred.

Recent research has also produced mechanisms for estimating other important statistics such as the total number of flows, the number of flows of size 1, and the first and second moments of the flow size distribution. However, such solutions are often restricted to producing estimates for only the complete population of flows. Since estimates of the SubFSD can be used to infer these statistics for the corresponding subpopulation, a valuable by-product of estimating SubFSD is the ability to estimate such derived statistics for arbitrary subpopulations.

FIGURE 10 illustrates an example of a network monitoring device 1000 that monitors a stream of data packets 22 over a link (not shown) of a communications network, according to one embodiment. The online streaming module 32, as explained above with respect to FIGURE 2, operates on all the packets. It is a simple array of counters, and does not retain any information about the packet headers 23, and cannot be used for estimating flow size distribution of the subpopulations by itself. On each packet arrival, the flow label is hashed to generate an index into this array and the corresponding counter is incremented. There is no attempt to detect or avoid collisions. As a result, several flows might increment the same counter due to collisions in hashing. At the end of the measurement epoch, the entire array is sent to the estimation module 97 or to storage.

The sampling module 38 only samples a small percentage of the traffic and retains information about only the sampled packets. The sampling module 38 maintains per-flow records for all sampled packets. At the end of each epoch, the flow label and flow size in packets from each record is summarized in a list and sent to the estimation module 97 or to storage.

The data collected by each module can be processed immediately at the end of the epoch, or at a much later time. The estimation module 97 has a preprocessing submodule 98 and an estimation submodule 99. The preprocessing submodule 98 takes the data collected by the online streaming module 32 and sampling modules 38, along with a specification of the subpopulation of interest, and produces a composite data structure of preprocessed data. The preprocessing submodule 98 uses a filter specifying the subpopulation of interest to synthesize the data from the online streaming module 32 and the sampling module 38. Since the sampling records contain enough data to reconstruct the flow labels of all the sampled flows, the hash function from the online streaming module 32 can be used to associate each sampling record with a unique index in the array of counters. Also, the subpopulation filter can be used to further classify the sampled flows as either belonging to the subpopulation S or its complement $\overline{S}$. The subpopulation filter can be a set of flow classification rules such that a flow matching any of the rules is defined to be in S and a flow matching none of the rules is in $\overline{S}$.

FIGURE 11 illustrates an overview of a method of estimating flow size distribution using an online streaming module and a sampling module, according to one embodiment. As explained earlier with respect to FIGURES 1-3, the method is run using software set up at

each link of interest in a network so that the packets crossing that link can be monitored. In 1107, the storage needs of the sketch are estimated. In 1110, the online streaming module 32 monitors the packet stream and generates a sketch for a data packet stream run over a link during a period of time. In 1111, the sampling module samples certain packets to obtain sampling information about those packets. In 1115, the sketch generated by the online streaming module 32 and the sampling information generated by the sampling module 38 are sent to a central server 44. In 1120, the central server 44 performs algorithms analyzing the sketch and the sampling information.

FIGURE 12 illustrates example output of the preprocessing module 98, according to one embodiment. The first counter has a value 1 but has no associated sampled flows, because the only packet hashing to this index was not sampled. The second counter has a value of 20 and is associated with two sampled flows, one from S with 3 sampled packets and the other from $\overline{S}$ with just one sampled packet. The third counter has a value of 2 and is associated with a single flow from S with one sampled packet. Because sampling misses a large percentage of packets, the list of samples associated with any counter does not provide a complete description of all the flows hashing to that counter. However, the sampled flow sizes do provide some information about the possible values of the actual flow sizes (before sampling), and more importantly, their membership in S or $\overline{S}$. The estimation submodule 99 exploits this information to compute accurate estimates of flow size distribution for S and $\overline{S}$. This preprocessed data is the input to the estimation submodule 99 that produces the final estimate of the specified subpopulation.

Note the difference between the flow label and the specification of a subpopulation of interest. The flow label refers to picking the fields of the header that will be stored for each sampled packed or hashed to pick an index in the counter array. The specification of the subpopulation of interest refers to a specific filter that defines a subpopulation. For example, the flow label can be defined as the four-tuple of source IP, destination IP, source port and destination port. A possible filter would be "source port = 80", thus specifying all webserver-to-client flows as the subpopulation of interest. The flow label needs to be defined at the beginning, whereas the filters specifying the various subpopulations can be chosen at query time.

FIGURE 13 is pseudocode that illustrates a method for estimating the SubFSD from observed data, according to one embodiment. FIGURE 13 finds the MLE using an EM. As

with the pseudocode of FIGURE 9, we start with a guess of the parameters we would like to estimate. Typically the expectations of the missing values can be computed conditioned on these parameters. Then we replace the occurrences of the missing values in the likelihood function $P(\Theta \mid \gamma)$ by their conditional mean. Finally, maximizing the likelihood function with these replacements results in a new and better estimate of the parameters. This process will iterate multiple times until the estimated parameters converge to a set of values (e.g., a local MLE).

Referring to FIGURE 13, $\theta = \{n, \phi, n', \phi'\}$ is estimated. The algorithm begins with an initial guess of Θ^{ini} (see line 2 of FIGURE 13), which is generated by using the observed distribution of counter values in the array of counters as the initial uses for both ϕ and ϕ'. The initial guesses for n and n' are generated by taking the total number of non-zero counters and partitioning this number in the ratio of the number sampled flows from S and $\overline{S}$ in the sampled data, respectively.

Each iteration of the algorithm proceeds as follows. For each of the m counters, we perform the following steps. Recall the v_i is the value of the counter at index i and q_i is the list of 2-tuples representing the sampled data associated with this counter. Let $\Omega_{(v_i,q_i)}$ represent the set of feasible split patterns for the counter at index i. Then, for each split pattern $\alpha \in \Omega_{(v_i,q_i)}$, we compute $P(\alpha \mid \Theta, v_i, q_i)$, the probability of occurrence of α conditioned upon the current estimate of Θ and the observations vi and qi associated with this counter. This probability is given by:

$$P(\alpha \mid \Theta, v, q) = \frac{\sum_{\pi \in \Pi(\alpha,q)} P(\pi \mid \alpha, w) P(\alpha \mid \Theta, v)}{\sum_{\beta \in \Omega(v)} \sum_{\pi \in \Pi(\beta,q)} P(\alpha \mid \beta, w) P(\beta \mid \Theta, v)}$$

where:

$$P(\pi \mid \alpha, w) = \frac{(\prod_{i=1}^{l} \binom{a_i}{b_i})(\prod_{j=1}^{r} \binom{a'_j}{b'_j})}{\binom{v}{w}}$$

corresponding to b_i packets being chosen from flows of size a_i respectively from S in the split pattern α, $i \in \{1, 2, \ldots, l\}$ and b'_j packets being chosen from flows of size a'_j respectively from $\overline{S}$ in α, $j \in \{1, 2, \ldots, r\}$. Also:

$$P(\alpha \mid \Theta, v) = \frac{P(\alpha \mid \Theta)}{\sum_{\beta \in \Omega(v)} P(\beta \mid \Theta)}$$

Where

$P(\alpha \mid \Theta)$

And

$P(\beta \mid \Theta)$

can be computed from the formula:

$$P(\alpha \mid \Theta) = \left(e^{-n/m} \prod_{i=1}^{k} \frac{(n_{q_i}/m)^{f_i}}{f_i!} \right) \left(e^{-n'/m} \prod_{i=1}^{k} \frac{(n'_{q'_i}/m)^{f'_i}}{f'_i!} \right)$$

where α is a split pattern in which there are f_1 *flows of size* s_1, f_2 *flows of size* s_2, ..., f_k *flows of size* t_k *from S, and* f'_1 *flows of size* s'_1, f'_2 *flows of size* s'_2, ..., f'_k *flows of size* s'_q *from* $\overline{S}$.

The contribution of this event α, weighted by this probability, is credited to the flow counts of all constituent flows in α. Suppose $\alpha = \{< a_1, S >, < a_2, S >, \ldots, < a_1, S >, < a'_1, \overline{S} >, < a'_2, \overline{S}' >, \ldots, < a'_r, \overline{S} >\}$, i.e., the split pattern α corresponds to the event that l flows from S with sizes a_1, a_2, .., a_1 and r flows from $\overline{S}$ with sizes a'_1, a'_2, ..., a'_r collide at the counter in question. Then we increment each of the counts $n'_{a_j}, j \in \{1, 2, \ldots, r\}$ by $P(\alpha \mid \Theta, v_i, q_i)$. This tabulation process is illustrated through the following example.

Say the value of a counter at some index i is $v_i = 2$, and the sampled flow set is $q_i = \{< 1, S >\}$. Then $\Omega_{(v_i, q_i)}$, the set of split patterns that are consistent with the observation contains three elements: (i) $\{< 2, S >\}$, (ii) $\{< 1, S >, < 1, S >\}$, and (iii) $\{< 1, S >, <1, \overline{S} >\}$. Note that other possible split patterns of 2, such as $< 2, \overline{S} >$, are excluded since they are not consistent with the observation $\{< 1, S >\}$. Suppose the respective probabilities of the events

(i), (ii), and (iii), are .5, .3, and .2. A probability of .5 for event (i) implies that, with this probability, exactly one flow of size 2 from subpopulation S hashed to counter i, and exactly one packet from this flow was sampled. To tabulate this, we should increment our count of n_2, the number of flows in S that have size 2, by .5. Similarly, the probabilities of events (ii) and (iii) are used to increment the counts of the constituent flows. In all, we execute the following three sets of operations, corresponding to these three events:

(i) $n_2 := n_2 + .5$;

(ii) $n_1 := n_1 + .3$; $n_1 := n_1 + .3$

(iii) $n_1 := n_1 + .2$; $n_1' := n_1' + .2$

Summarizing all 3 cases, we credit .5 to the counter n_2, .3 + .3 + .2 = .8 to the counter n_1, and .2 to the counter n_1'. Thus, at the end of the loop for counter i (line 5), the algorithm has considered all split patterns consistent with the observations and used the respective probabilities of each of these splits to increment the counts of the constituent flow sizes. After processing each index i $\in$ 1, 2, ..., m in this manner, we obtain a final tabulation of the counters for the flow counts $n_1, n_2, \ldots, n_z, n_1', n_2', \ldots, n_z'$, and obtain n^{new}, $(n')^{new}$. We then renormalize the counts into new (and refined) flow distribution ϕ^{new}, $(\phi')^{new}$.

Conclusion

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to persons skilled in the relevant art(s) that various changes in form and detail can be made therein without departing from the spirit and scope of the present invention. In fact, after reading the above description, it will be apparent to one skilled in the relevant art(s) how to implement the invention in alternative embodiments. Thus, the present invention should not be limited by any of the above described exemplary embodiments. In particular, it should be noted that, for example purposes, the above explanation has focused on the example(s) of [list the specific embodiment(s)]. However, those experienced in the art will realize that multiple other embodiments can be used.

In addition, it should be understood that any figures, screen shots, tables, examples, etc. which highlight the functionality and advantages of the present invention, are presented for example purposes only. The architecture of the present invention is sufficiently flexible and configurable, such that it may be utilized in ways other than that shown. For example, the steps listed in any flowchart may be re-ordered or only optionally used in some embodiments.

Further, the purpose of the Abstract of the Disclosure is to enable the U.S. Patent and Trademark Office and the public generally, and especially the scientists, engineers and practitioners in the art who are not familiar with patent or legal terms or phraseology, to determine quickly from a cursory inspection the nature and essence of the technical disclosure of the application. The Abstract of the Disclosure is not intended to be limiting as to the scope of the present invention in any way.

Furthermore, it is the applicant's intent that only claims that include the express language "means for" or "step for" be interpreted under 35 U.S.C. 112, paragraph 6. Claims that do not expressly include the phrase "means for" or "step for" are not to be interpreted under 35 U.S.C. 112, paragraph 6.

CLAIMS

1. A network monitoring system, comprising:

a collection unit that monitors a packet stream over an observation point in a network, the collection unit further comprising a sketch generator that generates a sketch data structure during a sketch period, the sketch data structure containing zero or more pieces of information about each data packet in the packet stream during the sketch period; and

an analysis unit that receives one or more sketches whose sketch period is within a measurement interval, the analysis unit further comprising a monitoring application that generates an estimate of flow size distribution components for the observation point during the measurement interval based on the one or more received sketches.

2. The system of claim 1, wherein the sketch further comprises a counter array sketch having a plurality of counters each capable of storing a count wherein the sketch generator further comprises a hash function that hashes an attribute of a flow label of each data packet to generate an index value wherein each index value corresponds to a counter of the counter array so that each counter contains a count of a number of data packets whose flow label attribute hashes to that index value that have been monitored at the collection unit, during the sketch period.

3. The system of claim 2, wherein the flow label attribute for each data packet further comprises one or more of a source address for the data packet, a destination address for the data packet, a source port for the data packet, a destination port for the data packet and a protocol for the data packet.

4. The system of claim 1, wherein the monitoring application further comprises an estimation module that estimates the number of flows of size 1 over the observation point.

5. The system of claim 1, wherein the monitoring application further comprises an estimation module that estimates the complete distribution of flow sizes over the observation point.

6. The system of claim 5, wherein the monitoring application accounts for collisions.

7. The system of claim 5, wherein high counter values ignore the collisions above a predetermined number of counts.

8. The system of claim 2, wherein the sketch further comprises a multi-resolution array of counters.

9. The system of claim 1, wherein the observation point is a link or node.

10. A method for monitoring a network, comprising:

monitoring the packet stream over an observation point in a network using a collection unit;

generating a sketch data structure during a sketch period, the sketch data structure containing zero or more pieces of information about each data packet in the packet stream during the sketch period;

receiving one or more sketches whose sketch period is within a measurement interval; and

generating a flow estimate for the observation point during the measurement interval based on the one or more received sketches.

11. The method of claim 10, wherein generating the sketch data structure further comprises generating a counter array sketch having a plurality of counters each capable of storing a count and hashing an attribute of a flow label of each data packet to generate an index value wherein each index value corresponds to a counter of the counter array so that each counter contains a count of a number of data packets whose flow label attribute hashes to that index value that has been monitored.

12. The method of claim 11, wherein the flow label attribute for each data packet further comprises one or more of a source address for the data packet, a destination address for the data packet, a source port for the data packet, a destination port for the data packet and a protocol for the data packet.

13. The method of claim 10, wherein the number of flows of size 1 over the observation point is estimated.

14. The method of claim 10, wherein the complete distribution of flow sizes over the observation point is estimated.

15. The method of claim 15, wherein collisions are accounted for.

16. The method of claim 15, wherein the collisions are ignored above a predetermined number of counts.

17. The method of claim 11, wherein the sketch further comprises a multi-resolution array of counters.

18. The method of claim 10, wherein the observation point is a link or node.

19. A network monitoring system comprising:

a collection unit that monitors a packet stream over a observation point in a network, the collection unit further comprising a sketch generator and a sampling module, the sketch generator generating a sketch data structure during a sketch period, the sketch data structure

containing one or more pieces of information about each data packet in the packet stream during the sketch period, the sampling module collecting portions of zero or more data packets in the packet stream during the sketch period; and

an analysis unit that receives one or more sketches whose sketch period is within a measurement interval and samples for the same period, the analysis unit further comprising a monitoring application that generates an estimate of flow size distribution components for the observation point during the measurement interval based on one or more of the received sketches and one or more of the received samples.

20. The system of claim 19, wherein the sketch further comprises a counter array sketch having a plurality of counters each capable of storing a count wherein the sketch generator further comprises a hash function that hashes an attribute of a flow label of each data packet to generate an index value wherein each index value corresponds to a counter of the counter array so that each counter contains a count of a number of data packets whose flow label attribute hashes to that index value that have been monitored at the collection unit.

21. The system of claim 20, wherein the flow label attribute for each data packet further comprises one or more of a source address for the data packet, a destination address for the data packet, a source port for the data packet, a destination port for the data packet and a protocol for the data packet.

22. The system of claim 19, wherein data packet sampling is performed periodically.

23. The system of claim 19, wherein the data packets are sampled based on random numbers drawn from Constant, Uniform, Poisson, Gaussian, and Pareto distributions and their combinations and approximations.

24. The system of claim 19, wherein the collection unit aggregates samples into flow records based on common fields in data packet headers.

25. The system of claim 19, wherein the monitoring application associates samples or aggregates thereof with various locations in the sketch.

26. The system of claim 19, wherein the association is performed using the same hash function as used in the sketch, the hash function operating on the same collection of packet header fields as in the sketch.

27. The system of claim 25, wherein a request for an estimate of flow size distribution components specifies a subpopulation of interest.

28. The system of claim 27, wherein the monitoring application further comprises an estimation module that estimates the total number of flows for the requested subpopulation.

29. The system of claim 27, wherein the monitoring application further comprises an estimation module that estimates the number of flows of size 1 over the observation point for the requested subpopulation.

30. The system of claim 27, wherein the monitoring application further comprises an estimation module that estimates the distribution of flow-sizes over the observation point for the requested subpopulation.

31. The system of claim 20, wherein the sketch further comprises a multi-resolution array of counters.

32. The system of claim 19, wherein the observation point is a link or node.

33. A method for monitoring a network, comprising:

monitoring a packet stream over an observation point in a network;

generating a sketch data structure during a sketch period, the sketch data structure containing one or more pieces of information about each data packet in the packet stream during the sketch period;

collecting portions of zero or more data packets in the packet stream during the sketch period;

receiving one or more sketches whose sketch period is within a measurement interval and samples for the same period; and

generating an estimate of flow size distribution components for the observation point during the measurement interval based on one or more of the received sketches and one or more of the received samples.

34. The method of claim 33, wherein generating the sketch data structure further comprises generating a counter array sketch having a plurality of counters each capable of storing a count and hashing an attribute of a flow label of each data packet to generate an index value wherein each index value corresponds to a counter of the counter array so that each counter contains a count of a number of data packets whose flow label attribute hashes to that index value that has been monitored.

35. The method of claim 34, wherein the flow label attribute for each data packet further comprises one or more of a source address for the data packet, a destination address for the data packet, a source port for the data packet, a destination port for the data packet and a protocol for the data packet.

36. The method of claim 33, wherein data packet sampling is performed periodically.

37. The method of claim 33, wherein the data packets are sampled based on random numbers drawn from Constant, Uniform, Poisson, Gaussian, and Pareto distributions and their combinations and approximations.

38. The method of claim 33, wherein samples are aggregated into flow records based on common fields in data packet headers.

39. The method of claim 33, wherein samples or aggregates thereof are associated with various locations in the sketch.

40. The method of claim 33, wherein the association is performed using the same hash function as used in the sketch, the hash function operating on the same collection of packet header fields as in the sketch.

41. The method of claim 33, wherein a request for an estimate of flow size distribution components specifies a subpopulation of interest.

42. The method of claim 39, wherein the total number of flows for the requested subpopulation is estimated.

43. The method of claim 41, wherein the number of flows of size 1 over the observation point for the requested subpopulation is estimated.

44. The method of claim 41, wherein the distribution of flow-sizes over the observation point for the requested subpopulation is estimated.

45. The method of claim 34, wherein the sketch further comprises a multi-resolution array of counters.

46. The method of claim 33, wherein the observation point is a link or node.

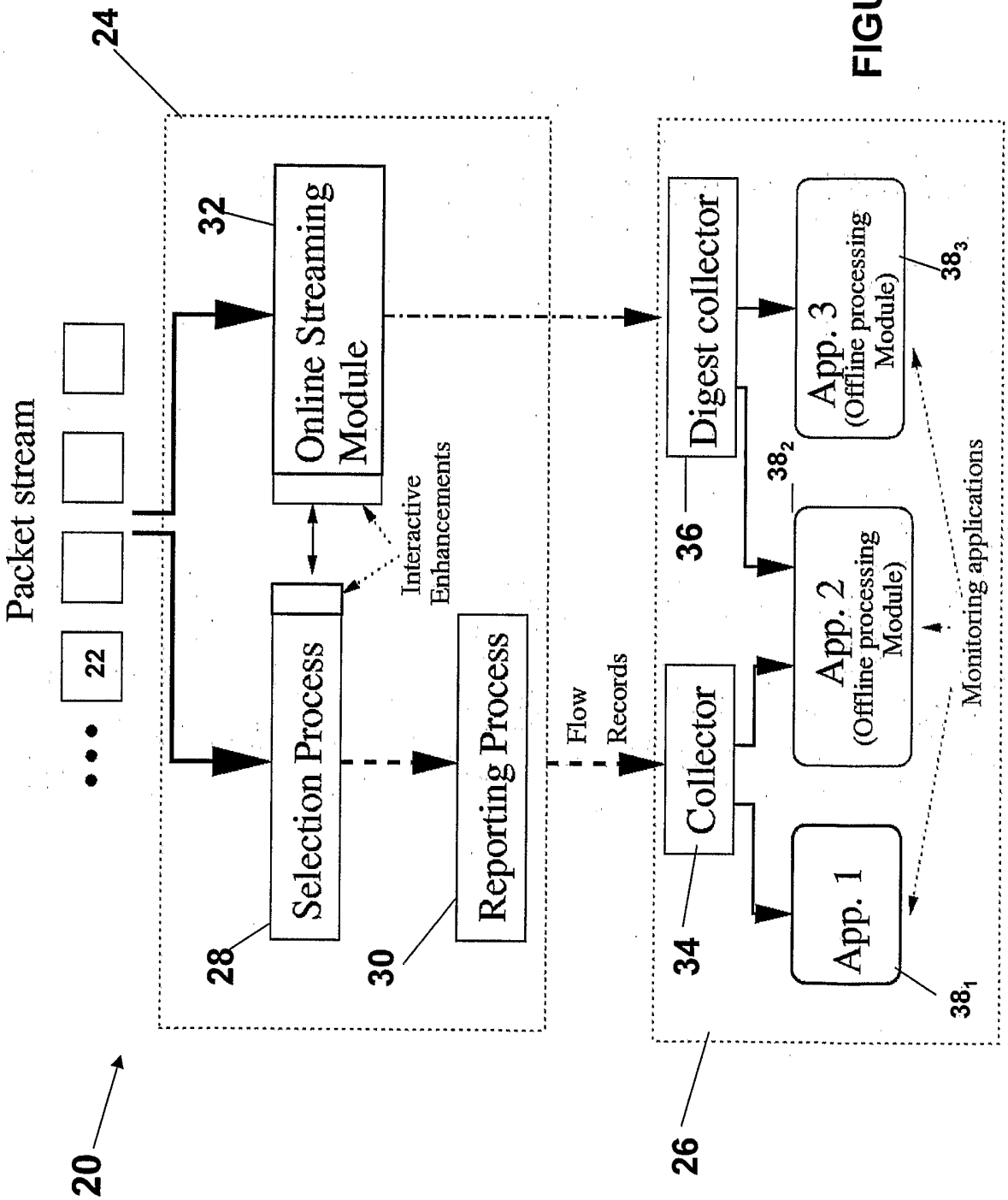
Packet stream
22
24
32
Online Streaming Module
Interactive Enhancements
Selection Process
28
Reporting Process
30
Flow Records
Digest collector
36
Collector
34
App. 1
App. 2 (Offline processing Module)
App. 3 (Offline processing Module)
38_1
38_2
38_3
Monitoring applications
20
26

FIGURE 1

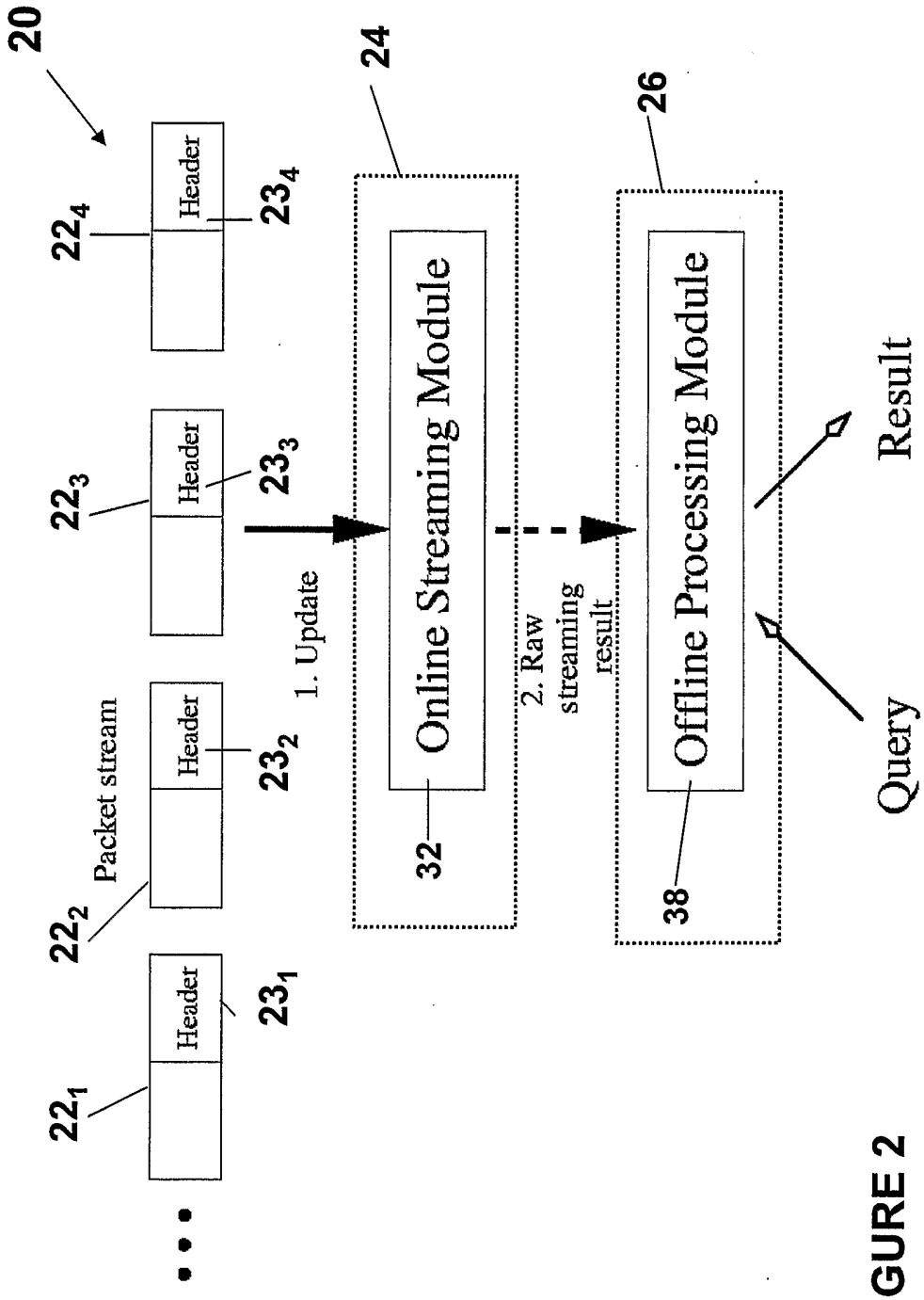
20
24
26
Header
Header
Header
Header
22_4
22_3
22_2
22_1
23_4
23_3
23_2
23_1
Packet stream
1. Update
Online Streaming Module
32
2. Raw streaming result
Offline Processing Module
38
Result
Query

FIGURE 2

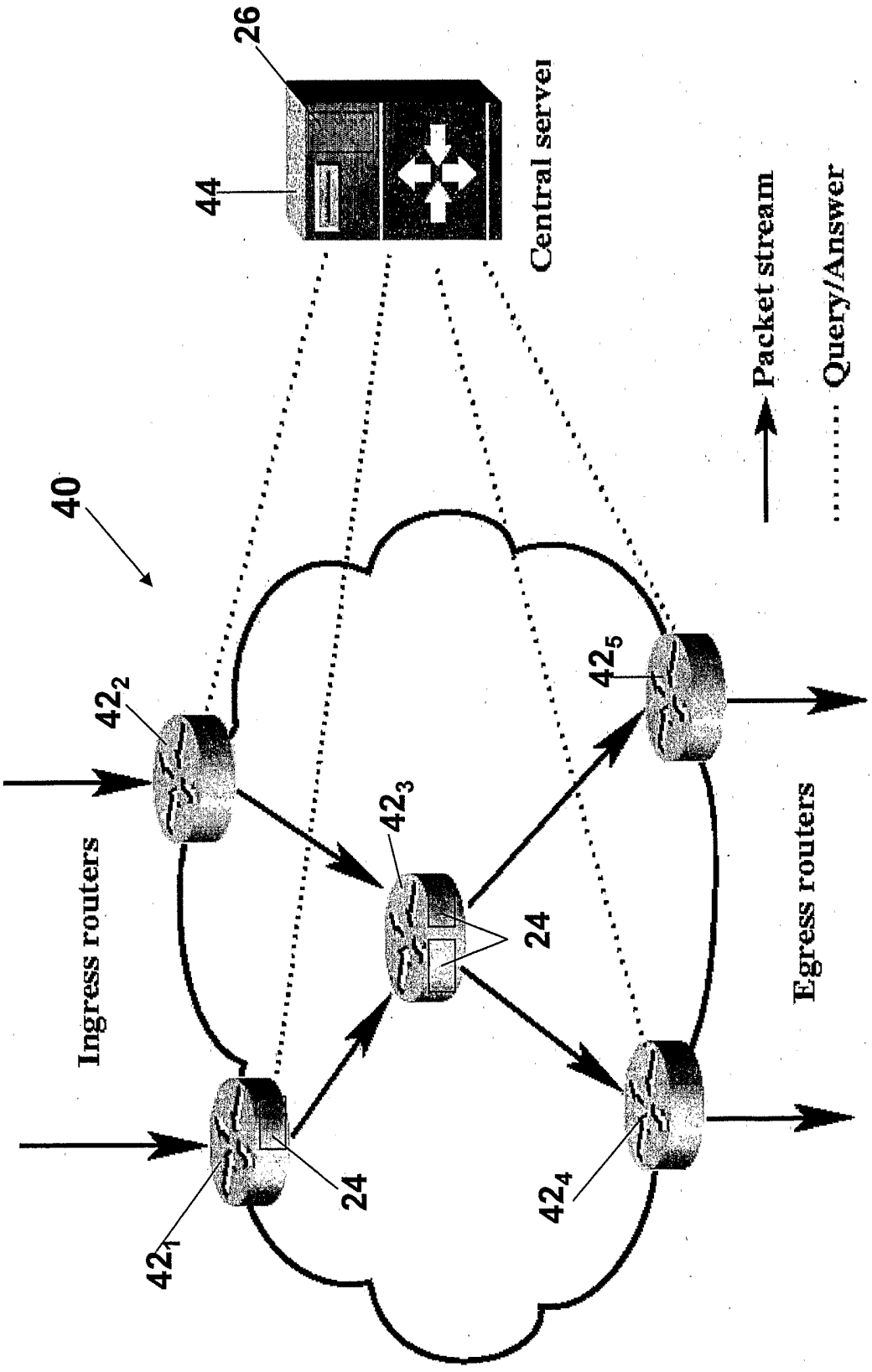
26
44
Central server
Packet stream
Query/Answer
40
42_2
42_5
42_3
24
Ingress routers
Egress routers
24
42_1
42_4

FIGURE 3

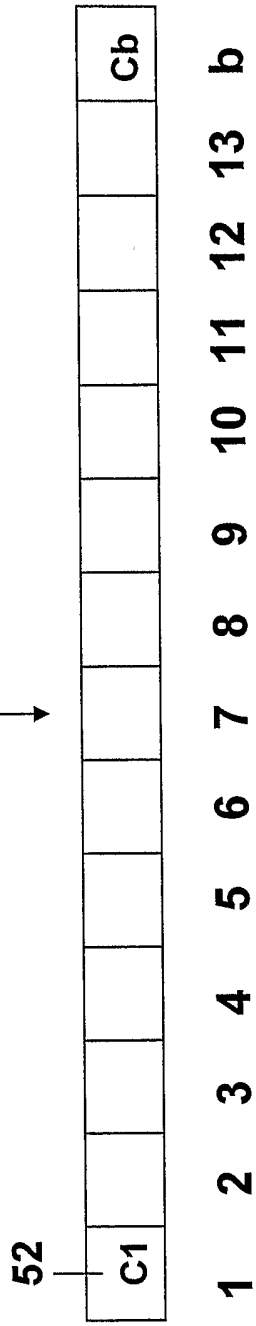

FIGURE 4A

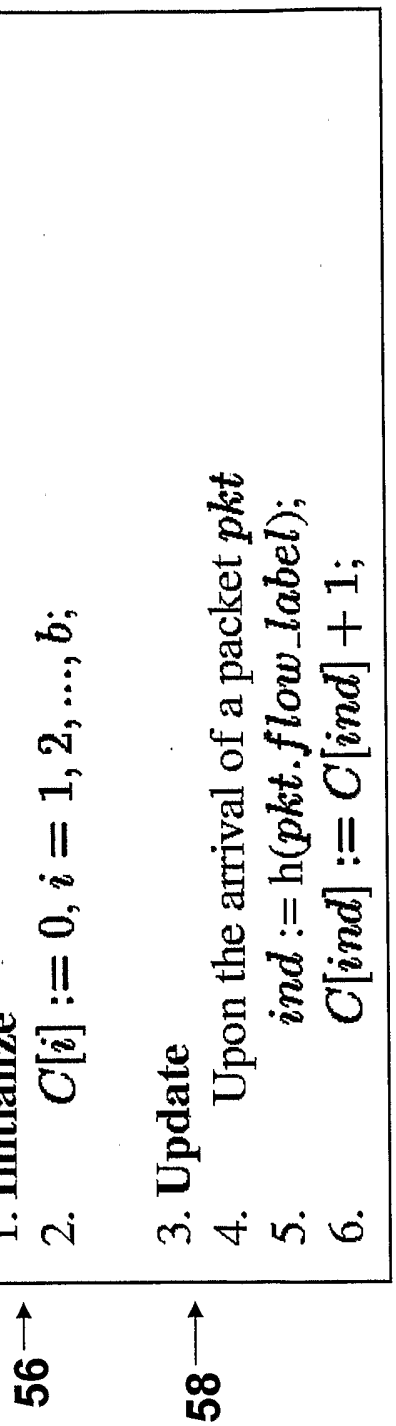

FIGURE 4B

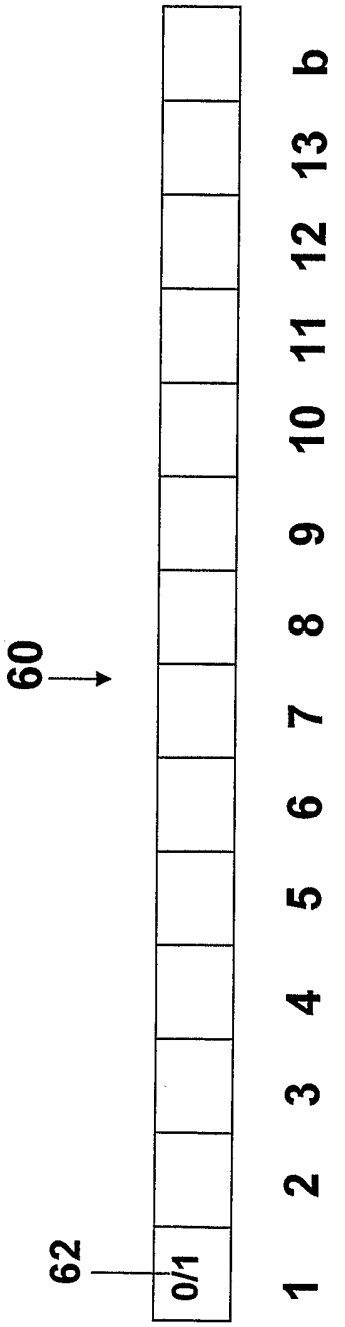

FIGURE 5A

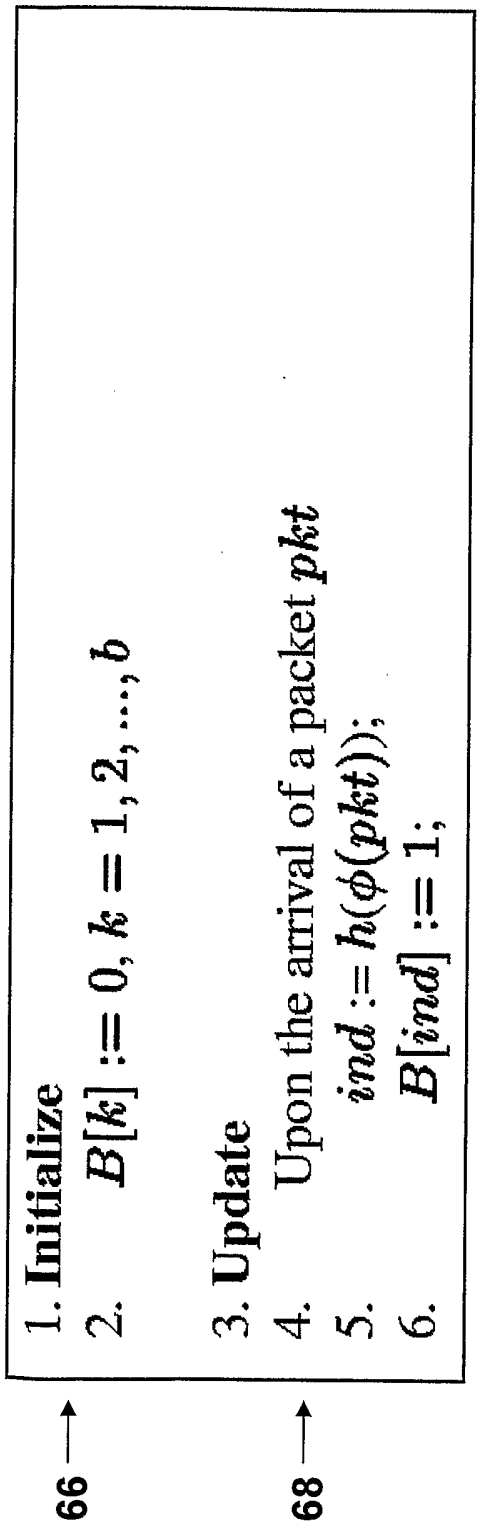

FIGURE 5B

FIGURE 6

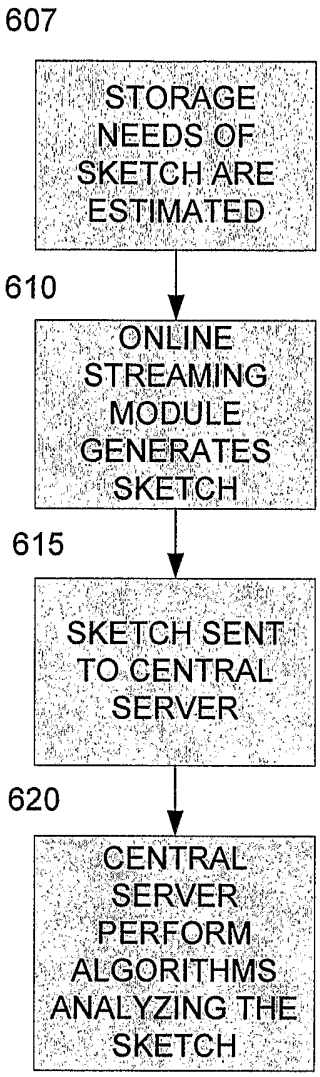
607
STORAGE NEEDS OF SKETCH ARE ESTIMATED
610
ONLINE STREAMING MODULE GENERATES SKETCH
615
SKETCH SENT TO CENTRAL SERVER
620
CENTRAL SERVER PERFORM ALGORITHMS ANALYZING THE SKETCH

FIGURE 7

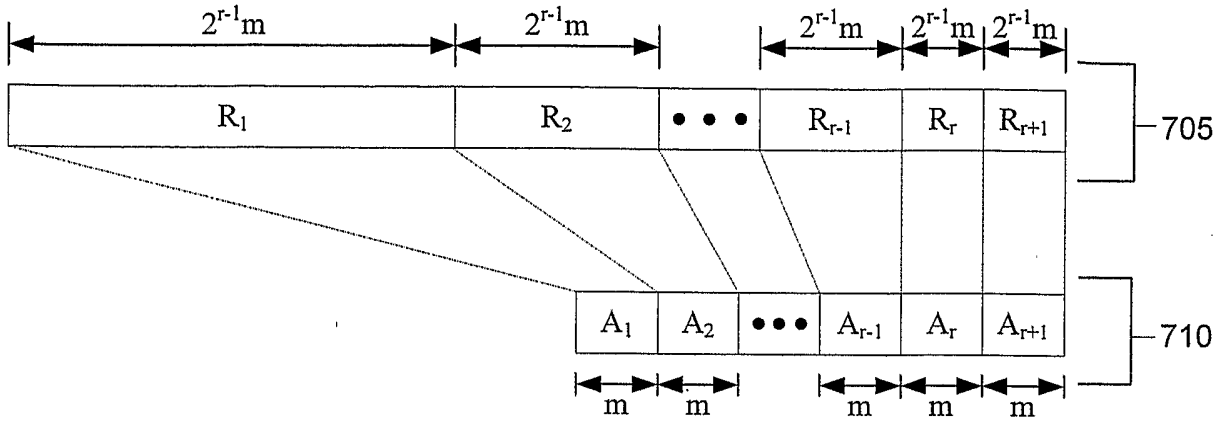
$2^{r-1}m$
$2^{r-1}m$
$2^{r-1}m$
$2^{r-1}m$
$2^{r-1}m$
R_1
R_2
R_{r-1}
R_r
R_{r+1}
705
A_1
A_2
A_{r-1}
A_r
A_{r+1}
710
m
m
m
m
m

FIGURE 8

1. **Initialize**
2. $r = \log_2(M/m)$
3. $R_i = \left\{ \begin{array}{ll} \left[(1-\frac{1}{2^{i-1}})M, (1-\frac{1}{2^i})M \right] & i = 1, 2, 3, \ldots, r \\ \left[(1-\frac{1}{2^r})M, M \right) & i = r+1 \end{array} \right\}$
4. Arrays $A_1, A_2, \ldots, A_{r+1}$ are all initialized to 0
5. **Update**
6. Upon arrival of a packet *pkt*
7. *ind* := hash(*pkt.flow_label*);
8. if (*ind* $\in R_j$)
9. $A_j[ind \bmod m]$++;

FIGURE 9

Input: y_i, number of counters have value i $(1 \le i \le z)$
Output: MLE for the flow distribution ϕ

1. Initialization: pick an initial flow distribution $\phi^{(ini)}$ and estimate the total flow count n^{ini} from Section 3.1.
2. $\phi^{new} := \phi^{ini}$; $n^{new} = n^{ini}$
3. **while** (convergence condition is not satisfied)
4. $\phi^{old} := \phi^{new}$; $n^{old} := n^{new}$
5. **for** $i := 1$ **to** z
6. **foreach** $\beta \in \Omega_i$
7. /*Ω_i is the set of all "collision patterns"
8. that add up to i, defined in Theorem 1*/
9. Suppose β is that f_1 flows of size s_1, f_2 flows of
10. size s_2, . . . , and f_q flows of size s_q collide into
11. a counter of value i, then
12. **for** $j := 1$ **to** q
13. $n_{s_j} := n_{s_j} + y_i * f_j * p(\beta \mid \phi^{old}, n, \mathrm{V} = i)$
14.
15.
16. **end**
17. **end**
18. **end**
19. $n^{new} := \sum_{i=1}^{z} n_i$
20. **for** $i := 1$ **to** z
21. $\phi^{new} := n_i / n^{new}$
22. **end**
23. /* normalize the counts $n_i's$ into flow distribution ϕ */.
24. **end**

FIGURE 10

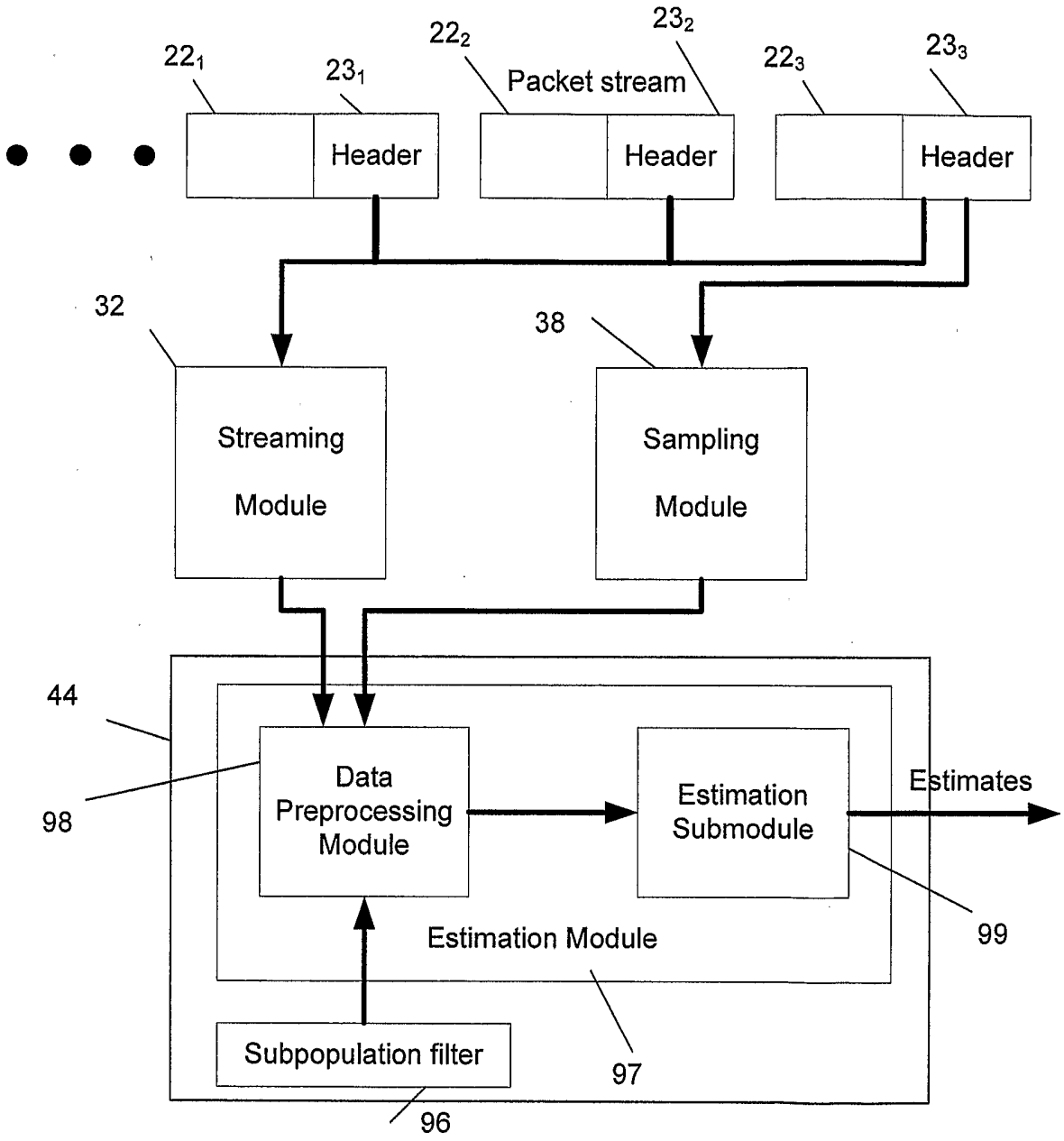
22_1
23_1
22_2
23_2
Packet stream
22_3
23_3
Header
Header
Header
32
38
Streaming Module
Sampling Module
44
98
Data Preprocessing Module
Estimation Submodule
Estimates
Estimation Module
99
Subpopulation filter
97
96

FIGURE 11

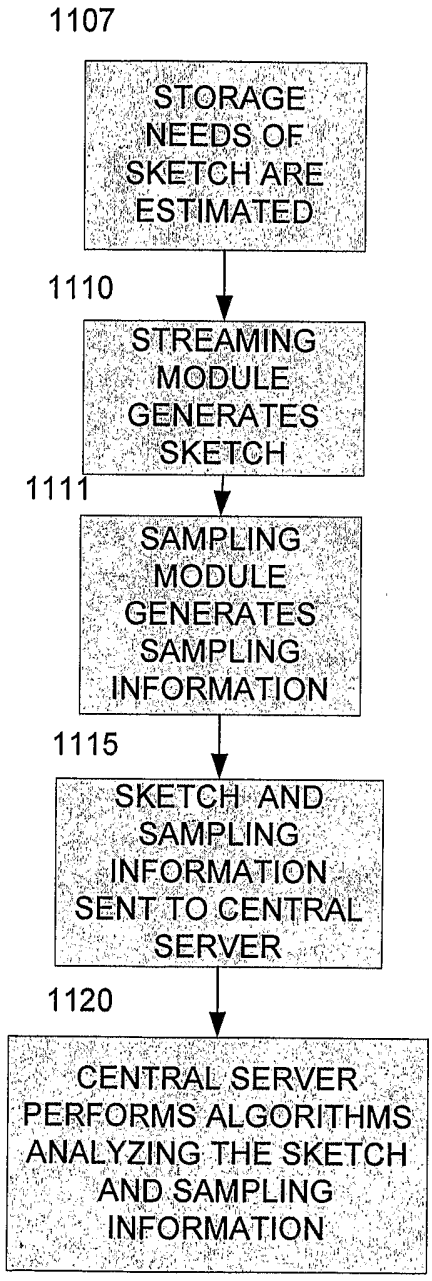
1107
STORAGE NEEDS OF SKETCH ARE ESTIMATED
1110
STREAMING MODULE GENERATES SKETCH
1111
SAMPLING MODULE GENERATES SAMPLING INFORMATION
1115
SKETCH AND SAMPLING INFORMATION SENT TO CENTRAL SERVER
1120
CENTRAL SERVER PERFORMS ALGORITHMS ANALYZING THE SKETCH AND SAMPLING INFORMATION

FIGURE 12

Index (i)	Counter value (v_i)	list of sampled flow sizes from S and $\overline{S}$ (q_i)
1	1	-
2	20	$<3, S>, <1, \overline{S}>$
3	2	$<1, S>$
4	1	$<1, \overline{S}>$
5	0	-
6	5	$<1, S>$
.	.	.
.	.	.
.	.	.

FIGURE 13

Input: our observation $\mathcal{Y} = \{< v_i, q_i >\}_{1 \le i \le m}$
Output: (local) MLE for Θ

1. Initialization: pick initial distribution Θ^{ini}
   as described in Section 4.3.3.
2. $\Theta^{new} := \Theta^{ini}$
3. **while** (convergence condition is not satisfied)
4. $\Theta^{old} := \Theta^{new}$;
5. **for** $i := 1$ **to** m
   /* $\Omega_{(v_i,q_i)}$ is the set of all split patterns
   consistent with the observation $< v_i, q_i >$ */
6. **foreach** $\alpha \in \Omega_{(v_i,q_i)}$
   /* Suppose $\alpha = \{ < a_1, S >, < a_2, S >, \ldots, < a_l, S >,$
   $< a'_1, \overline{S} >, < a'_2, \overline{S} >, \ldots, < a'_r, \overline{S} > \}$ */
7. for $j := 1$ to l
8. $n_{a_j} := n_{a_j} + P(\alpha|\Theta, v_i, q_i)$
9. for $j := 1$ to r
10. $n_{a'_j} := n_{a'_j} + P(\alpha|\Theta, v_i, q_i)$
11. **end**
12. **end**
13. **end**
    /* Compute the new estimate of n and n'. */
14. $n^{new} := \sum_{i=1}^{z} n_i$
15. $(n')^{new} := \sum_{i=1}^{z} (n'_i)$
    /* Normalize the flow counts n_i, n'_i into ϕ and ϕ',
    and clear the counter values for the next iteration. */
16. for $i := 1$ to z
17. $\phi_i^{new} := n_i / n^{new}$
18. $(\phi'_i)^{new} := n'_i / (n')^{new}$
19. $n_i := 0$
20. $n'_i := 0$
21. **end**
22. **end**