Reducing Memory in High-Speed Packet Classification

Viktor Puš
CESNET, a.l.e.
Zikova 4, 160 00 Prague 6, Czech Republic
Email: pus@cesnet.cz

Jan Kořenek
IT4Innovations Centre of Excellence
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, 612 66 Brno, Czech Republic
Email: korenek@fit.vutbr.cz

***Abstract*—Many packet classification algorithms were proposed to deal with the rapidly growing speed of computer networks. Unfortunately all of these algorithms are able to achieve high throughput only at the cost of excessively large memory and can be used only for small sets of rules. We propose new algorithm that uses four techniques to lower the memory requirements: division of rule set into subsets, removal of critical rules, prefix coloring and perfect hashing. The algorithm is designed for pipelined hardware implementation, can achieve the throughput of 266 million packets per second, which corresponds to 178 Gb/s for the shortest 64 B packets, and outperforms older approaches in terms of memory requirements by 66 % in average for the rule sets available to us.**

***Keywords*—FPGA, SRAM, hardware, parallelism, classification**

I. Introduction

The growth of computer networks provides more opportunities for new applications and services, but also gives new possibilities for suspicious activities. Malicious traffic is usually detected by Intrusion Detection Systems (IDS) and then filtered by firewalls which perform per-packet classification based on the given set of rules.

Packet classification (not to be confused with application-level traffic classification) is a problem of assigning each network packet into one or more classes. Classes are unambiguously determined by rules. Each rule defines a condition for all significant packet header fields (or *dimensions*). These fields are usually a 5-tuple (Source IP Address, Destination IP Address, Source Port, Destination Port, Protocol). A condition may be exact match, prefix match (usually for IP addresses), range match (for ports), or a wildcard, which matches any value. The goal of a packet classification algorithm is to find the matching rule with the highest priority. The output of the algorithm is the number of the matched rule.

As network speeds are increasing, the demand for a high speed packet classification is also growing. Usually, constant time complexity is required in order to achieve wire speed processing and avoid vulnerability to attacks. Classification algorithms commonly use preprocessing of rules to create a data structure that supports high-speed searching. Current algorithms and hardware architectures can achieve multigigabit speeds only at the cost of high memory requirements.

The first memory reduction method for high-speed packet classification was introduced in Multi Subset Crossproduct Algorithm (MSCA) [1], which reduces memory requirements by dividing the rule set into several subsets and by moving the worst rules into separate algorithm branch. Other methods were introduced in Prefix Coloring Classification Algorithm (PCCA) [2], which saves memory by additional prefix filtering and by using perfect hash function to map prefixes directly into the rule table. We evaluate both algorithms in terms of throughput and memory. Both algorithms aim at high-speed networks (tens or hundreds gigabits per second) and try to keep the memory requirements bounded.

Both algorithms work well for different rule sets, but none of them seems to produce small memory structures for all rule sets. Therefore we propose new algorithm which combines four reduction methods: division of rule set into subsets, removal of critical rules, prefix coloring and perfect hashing. All four methods combined in single algorithm outperform older approaches in terms of memory requirements by 66 % in average for the rule sets available to us. The algorithm is designed for pipelined hardware implementation and is able to process

266 million packets per second. The proposed algorithm is faster than MSCA and has the same throughput as PCCA.

The rest of the paper is organized as follows: The next section briefly introduces both of the discussed algorithms and explains the basic concept of pseudorules. Section III presents the new algorithm created as a combination of both. The following Section IV presents the obtained results. Section V concludes the paper.

II. Related work

The simplest classification scheme uses only one packet header field. Packet routing in IP networks is a common example of one-dimensional classification (only destination IP address is important for routing). This search on prefixes is called the *Longest Prefix Match* (LPM) operation. From the given set of prefixes with various lengths, the LPM algorithm finds the one that best fits the given full-length value. This corresponds to both IPv4 and IPv6 addressing schemes. Because the LPM operation is performed in IP packet routing, it is well studied topic. Basic algorithm and data structure for the LPM is a trie – the tree algorithm processing one input bit at each tree level and returning the last valid prefix visited. Trie is often modified to process more input bits in each step and to reduce memory requirements. Popular examples of such algorithms are the Tree Bitmap [3] and the Shape Shifting Trie [4], both having a strong potential for hardware implementation. Figure 1 shows an example of trie data structure for the longest prefix match operation. Black circles represent valid prefixes (possible results of LPM operation).

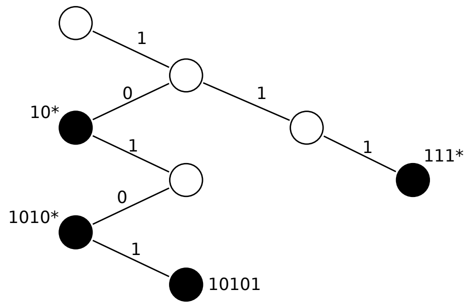

Fig. 1. Example of trie with four stored prefixes. Result for input 10100 will be 1010*.

Many algorithms [5], [6], [7] has been published for classification in multiple fields, but here we discuss only decomposition-based methods [1], [2], [8], [9], which may have constant time complexity of the search operation and are able to achieve the 100 Gb/s throughput. In decomposition methods, packet classification is divided into several steps, which can be implemented in hardware by pipeline stages. Figure 2 shows the basic scheme of decomposition algorithms.

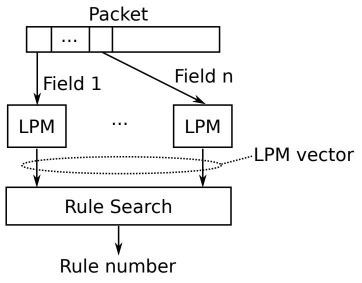

Fig. 2. Basic scheme of decomposition algorithms.

The input of packet classification is a vector of packet header fields. LPM operation is a first step which is performed for every packet header field independently. Range conditions (such as port ranges) in the ruleset are converted to prefixes, so that the LPM may be performed in all dimensions [1]. LPM is well studied and can be performed very fast: recently published approaches achieve billions of lookups per second using a dedicated hardware (ASIC or FPGA) [10]. Each LPM search engine returns one item from the *prefix set*, where prefix set is the set of all LPM results for the given ruleset and the given dimension. Result of the LPM Stage is the *LPM vector*, containing one prefix for each dimension. After the LPM, all fields of the resulting LPM vector must be processed in some way (this is specific for each algorithm) to get the correct rule number. Key issue is that the state space of LPM vectors can be extremely large. This is because all possible values of the LPM vector are obtained by creating the Cartesian product of prefix sets.

A. Pseudorules

Dharmapurikar et al. in [1] introduce the concept of *pseudorules*, which is fundamental in recent decomposition algorithms. We describe in detail how pseudorules are created and why this process increases the size of data structures in packet classification algorithms.

To cover all valid combinations of LPM results, pseudorules must be added to the ruleset. In fact, a pseudorule is always a special case of some rule. This is best explained by the example of pseudorules generation in Figure 3. We can see a simplified classification in two three-bit dimensions with three rules. In each dimension, trie is shown to illustrate the LPM operation. Colored arcs are the rules.

For example, LPM vector for packet with header fields

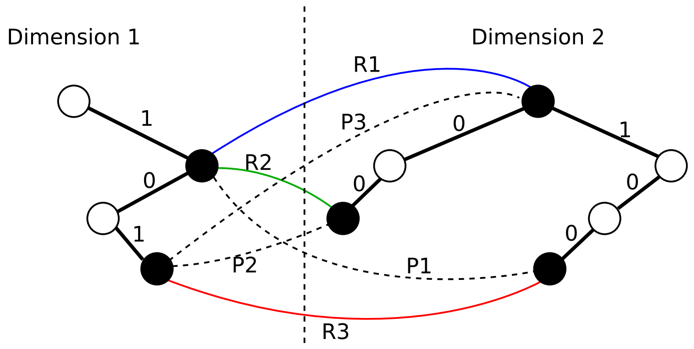

Fig. 3. Three rules $R1, R2, R3$ and three added pseudorules.

Rule	Dimension 1	Dimension 2	Target rule
R1	1*	*	R1
R2	1*	00*	R2
R3	101	100	R3
P1	1*	100	R1
P2	101	00*	R2
P3	101	*	R1

TABLE I
RULES AND PSEUDORULES.

$(111, 100)$ will be $(1*, 100)$. This combination is not in the original ruleset, but it is clear that the correct result is rule $R1(1*, *)$[1]. Therefore, pseudorule $P1(1*, 100)$ has to be added to handle this situation. Table I contains all rules and pseudorules together. *Target rule* in this table points to the correct classification result.

The generation of pseudorules is similar to Cartesian product, and may potentially expand the ruleset significantly, but not all possible combinations of prefixes need to be added. If the *universal rule* (a rule covering all possible packets) was in the ruleset, then all possible combinations would have to be added. However, this rule can be removed from the ruleset and returned as a result only if no other rule matches the packet. Therefore, pseudorules are a subset of Cartesian product of all prefix sets.

B. Multiple Subset Construction

Because pseudorules expansion is still similar to Cartesian product, MSCA [1] provides heuristics on how to break ruleset into several subsets, eliminating the majority of pseudorules. The algorithm exploits the fact that the sum of Cartesian products of small sets is much smaller than the single Cartesian product of large sets.

[1]Symbol * denotes prefix or wildcard

The paper also identifies rules that generate excessive amount of pseudorules. These rules are called *spoilers* and are treated in a separate algorithm branch to further reduce number of pseudorules. The LPM operation is slightly modified to return a result for each subset, because subsets may contain different prefixes. A Bloom filter [11] is associated with each subset to perform set membership query. If the Bloom filter output is true, one rule table memory access is performed to retrieve the resulting rule or pseudorule. An optimization technique is used to avoid single packet match several subsets, which would slow the algorithm down.

MSCA is designed for hardware implementation to achieve high throughput. The proposed architecture employs an FPGA and a single external SRAM. Figure 4 shows the basic structure of the MSCA (spoilers branch is not shown). Almost all parts of the algorithm are implemented in the FPGA, only the Rule Table is stored in the external memory. The Rule Table is the hash table of rules and remaining pseudorules. Given the fact that one rule is stored at hundreds of bits, the algorithm throughput is limited by the external memory bandwidth.

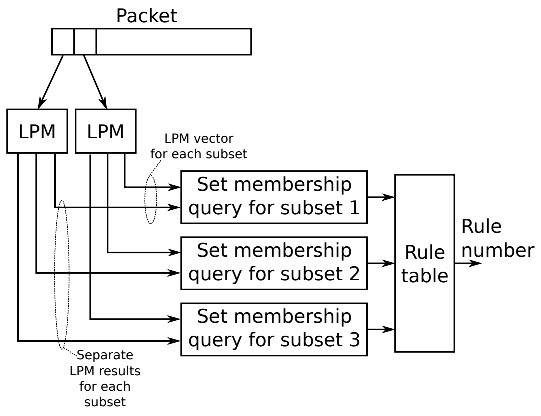
Packet
LPM
LPM
LPM vector for each subset
Set membership query for subset 1
Set membership query for subset 2
Set membership query for subset 3
Rule table
Rule number
Separate LPM results for each subset

Fig. 4. Hardware structure of the MSCA.

C. Perfect Hashing with Prefix Coloring

PCCA [2] notes the fact that the LPM operation is performed independently for each field in decomposition-based packet classification algorithms (see Figure 2). The advantage of this scheme is the strong potential for parallel computation. On the other hand, LPM results are logically related – only certain combinations of LPM results form a rule, the rest of them are unwished pseudorules. Thus, the knowledge of LPM result from one dimension *should* affect LPM result in other dimensions.

The LPM operation in PCCA is modified to return all matching prefixes, not only the longest one. Each prefix P stores its *color* and a precomputed bitmap for each of remaining dimensions. Each bit in the bitmap corresponds to one color. The bit corresponding to prefix R is set to 1 if prefixes P and R appear together in some rule. Otherwise, the bit is set to 0. This way, it is possible to remove most pseudorules easily by a simple logic, because each LPM result contains an information about "allowed" and "suppressed" prefixes from other dimensions.

PCCA also uses specifically constructed perfect hash function to map all pseudorules (in the form of LPM vectors) onto correct rules. This way, it is no longer necessary to store pseudorules, which saves a considerable amount of memory. Number of remaining pseudorules however still affects the memory of the algorithm, because the implementation of perfect hash function grows linearly with the number of keys (rules+pseudorules).

The perfect hash function consists of two ordinary hash functions which are used to compute two pointers to the vertex table. Filling of the vertex table includes construction of the random acyclic graph, where edges are hash keys and vertexes are output of the two ordinary hash functions. Only two integers are read from the vertex table when computing the perfect hash function.

Similarly to MSCA, PCCA also targets the implementation in FPGA and SRAM (Figure 4). Thanks to the fact that no pseudorules are stored in PCCA, the rule table can be stored in the FPGA. The SRAM is used to store the vertex table required by the hash function. Only two 16-bit integers are read from the SRAM each packet. This significantly lowers the required external memory bandwidth.

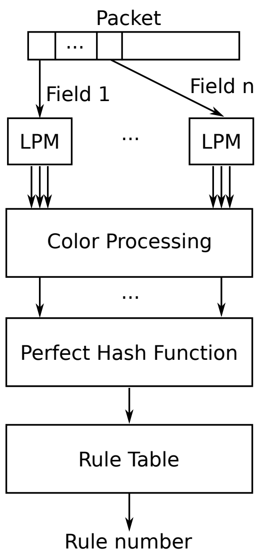

Fig. 5. Hardware structure of the PCCA.

III. Algorithm

In this chapter we introduce the new packet classification algorithm. Our aim is to achieve throughput at least 100 Gb/s for the shortest packets and to reduce the memory in comparison to MSCA and PCCA. We explore the possibility to combine both approaches into single algorithm which we call Multi Subset Prefix Coloring Classification Algorithm (MSPCCA).

The memory optimization techniques introduced in MSCA are orthogonal to the techniques used in PCCA This leads us to the idea of combining all methods into one compound algorithm. MSPCCA uses all four techniques:

- Division of the rule set into subsets.
- Removal of spoilers into separate algorithm branch.
- Prefix coloring and subsequent filtering.
- Perfect hash function construction.

The classification engine has the structure shown in Figure 6. (Spoilers branch is not shown.) Upon reception of each packet, the LPM is computed over all fields. The LPM results are available for each subset, therefore the processing is then split into separate independent branches. The following blocks therefore can run in parallel. The Color Processing blocks filter out impossible prefix combinations and outputs the LPM vectors. A Bloom filter for each subset is then queried for the presence of the LPM vector in the respective subset. Only in the case of positive result the Perfect Hash Function is computed to obtain the pointer to the rule table. The packet is then matched to the selected rule. In case of match, the selected rule is the output, otherwise the default rule is applied. In parallel to the main algorithm there is separate branch for classification of spoilers. This branch can be implemented as a small on-chip TCAM. The very end of the algorithm performs simple priority resolution between the two branches.

In parallel pipelined hardware implementation, several further optimizations are possible, compared to Figure 6. Supposing that MSCA is able to avoid match of single packet in multiple subsets, only one instance of the Perfect Hash Function logic is needed. The vertex table used by perfect hash function is separate for each subset. The vertex table is also the only part of the algorithm that is stored in the external memory. The perfect hash function reads two 16-bit integers from the external memory for each packet.

Several data structures must be precomputed before the packets can be classified by the algorithm. MSPCCA therefore works in two phases – precomputation and

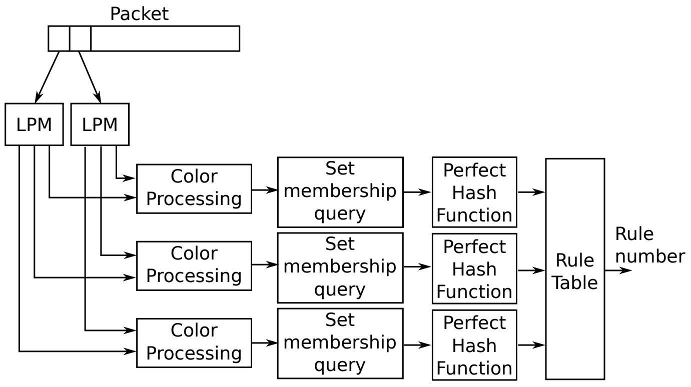

Fig. 6. Logical structure of the MSPCCA.

the classification itself. The precomputation phase is composed of separate methods as follows: After removing spoilers, MSCA splitting algorithm is used to split the rule set into subsets. Then for each subset, PCCA coloring algorithm is used to assign colors to prefixes and to create the reduced set of pseudorules. The Bloom filters are then filled with the rules and pseudorules of each subset. The last precomputation step is building the perfect hash function for each subset.

The first contribution of MSPCCA is the fact that all optimization techniques are successfully integrated into single algorithm. Spoilers removal, division into subsets and color processing contribute to lower the number of pseudorules, while the perfect hash function avoid storing the pseudorules. Therefore we expect the new algorithm to require less memory than its predecessors.

The second contribution is that the external memory bandwidth is defined by the *faster* of the two original algorithms – PCCA. Supposing that MSCA is able to avoid match of single packet in multiple subsets, only two 16-bit accesses to the external memory are required to classify a packet. The algorithm speed is not limited by MSCA.

IV. Results

A. Throughput

The algorithm is designed as a sequence of simple steps which can be directly implemented as a parallel hardware pipeline. We do not evaluate the throughput of LPM, because MSCA, PCCA and MSPCCA do not propose any new LPM algorithm. We just suppose that the constant number of bits in packet header fields leads to the constant time of LPM operation [10], [12].

The PCCA color processing uses only simple logic operations. Implementation of the Color Processing stage in Virtex-6 FPGA logic consumes 1364 LUT-FlipFlop pairs, and runs at 262 MHz (after synthesis for 5 dimensions and 8 colors). It only adds four cycles of latency. This small logic can also be easily replicated to achieve higher throughput if necessary.

The set membership query using Bloom filter requires the computation of several hash function and access to several bit locations in the on-chip memory. The suitability of implementation of Bloom filters is hardware was shown in many works before [1], [12], [13].

The perfect hash function computation requires computing two ordinary hash functions, two accesses to the external memory and a summation. All of these steps are simple, however the external memory limits the algorithm throughput. We suppose the use of RLDRAM2 running at 533 MHz [14] as external memory to store the vertex table. The algorithm throughput is 266 million packets per second, same as PCCA. Compare to 150 million packets per second throughput of 100 Gbps Ethernet for the shortest 64 B frames. The algorithm throughput is independent on the number of rules and the characteristics of the network traffic.

B. Memory

We compare the memory requirements of the new MSPCCA to the two original algorithms. We use two synthetic rule sets generated by ClassBench [15] (synth1-2) and four real-life rule sets from the university campus network (rules1-4). The rule sets are characterized in Table II. Numbers of rules and numbers of unique prefixes in each dimension are shown.

Rule set	Rules	Source IPs	Dest. IPs	Proto	SRC Ports	DST Ports
synth1	219	55	53	1	14	1
synth2	394	65	57	1	14	1
rules1	103	28	48	4	6	40
rules2	173	84	84	3	1	16
rules3	275	46	64	3	1	22
rules4	1 107	158	80	4	1	56

TABLE II
BASIC PROPERTIES OF RULE SETS.

Table III shows the memory of the algorithms for all six rule sets. The memory of the LPM is not considered. However, all three algorithms require to add some additional information to prefixes. This memory is included in the table. Eight spoilers are removed in each algorithm. For MSCA and MSPCCA, three subsets are used and the probability of Bloom filter false positive

Rule set	MSCA	PCCA	MSPCCA
synth1	1 622.91	227.51	122.91
synth2	2 928.36	601.23	232.91
rules1	2 303.81	3 347.68	82.96
rules2	162.26	1 075.99	81.45
rules3	211.68	260.02	122.41
rules4	898.77	814.27	568.66

TABLE III
MEMORY SIZE (KBITS) OF THE ALGORITHMS.

is set to 0.005. Eight colors are used in PCCA and MSPCCA.

It can be seen that MSCA has poor results for rule sets synth1, synth2 and rules1, while PCCA does not perform well for rules1 and rules2. The combined algorithm has smaller memory requirements for all rule sets. Also the algorithm stability is improved. While MSCA requires 722 to 22 367 and PCCA 645 to 32 501 bits per rule, MSPCCA stores only 445 to 805 bits per rule.

V. CONCLUSION

We have proposed the new high-speed packet classification algorithm which significantly reduces memory requirements. The algorithm is designed for pipelined hardware implementation and uses four memory reduction techniques: spoilers removal, rule set division, color filtering and perfect hashing. As can be seen in results, the proposed algorithm reduces memory from 36 to 96 % over MSCA and 30 to 97 % over PCCA. The average improvement is 66 % for the rule sets available to us.

The algorithm uses perfect hash function to look-up classification rule and need only two accesses to the hash table for every packet. Therefore high throughput can be achieved even for a large rule set, where off-chip memory is needed to store the hash table. Considering pipelined hardware architecture implemented in an FPGA and one off-chip memory, the algorithm analysis shows the throughput of 266 million packets per second, which corresponds to 178 Gb/s for the shortest 64 B packets and 548 Gb/s for the 440 B packets (reported as average in [16]). Moreover, the algorithm can not be overwhelmed by attacker, because the packet rate is independent on the number of rules and characteristics of the network traffic.

ACKNOWLEDGMENT

This work is supported by the "CESNET Large Infrastructure" project no. LM2010005 funded by the Ministry of Education, Youth and Sports of the Czech Republic, the research programme MSM 0021630528, the grant BUT FIT-S-11-1, the grant VG20102015022 and the IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

REFERENCES

[1] S. Dharmapurikar, H. Song, J. Turner, and J. Lockwood, "Fast packet classification using Bloom filters," in *ANCS '06: Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*. New York, NY, USA: ACM, 2006, pp. 61–70.

[2] V. Puš, M. Kajan, and J. Kořenek, "Hardware architecture for packet classification with prefix coloring," in *IEEE Design and Diagnostics of Electronic Circuits and Systems DDECS'2011*. IEEE Computer Society, 2011, pp. 231–236.

[3] W. Eatherton, G. Varghese, and Z. Dittia, "Tree bitmap: hardware/software IP lookups with incremental updates," *SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 97–122, 2004.

[4] H. Song, J. Turner, and J. Lockwood, "Shape shifting tries for faster ip route lookup," in *ICNP '05: Proceedings of the 13TH IEEE International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 358–367.

[5] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in *Proc. Hot Interconnects*, 1999.

[6] T. V. Lakshman and D. Stiliadis, "High-speed policy-based packet forwarding using efficient multi-dimensional range matching," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 203–214, 1998.

[7] D. Taylor and J. Turner, "Scalable packet classification using distributed crossproducing of field labels," in *IEEE INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, July 2005, pp. 269–280.

[8] V. Srinivasan, G. Varghese, S. Suri, and M. Waldvogel, "Fast and scalable layer four switching," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 191–202, 1998.

[9] V. Puš and J. Kořenek, "Fast and scalable packet classification using perfect hash functions," in *FPGA '09: Proceedings of the 17th international ACM/SIGDA symposium on Field programmable gate arrays*. New York, NY, USA: ACM, 2009.

[10] H. Lee, W. Jiang, and V. K. Prasanna, "Scalable High-Throughput SRAM-Based Architecture for IP Lookup Using FPGA," in *FPL '08*. IEEE, 2008.

[11] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, 1970.

[12] S. Dharmapurikar, P. Krishnamurthy, and D. E. Taylor, "Longest prefix matching using Bloom filters," in *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2003, pp. 201–212.

[13] Y. Lu, B. Prabhakar, and F. Bonomi, "Perfect hashing for network applications," *Information Theory, 2006 IEEE International Symposium on*, pp. 2774–2778, July 2006.

[14] "Rldram part catalog," Micron Technology, Inc.

[15] D. E. Taylor and J. S. Turner, "Classbench: a packet classification benchmark," *IEEE/ACM Trans. Netw.*, vol. 15, no. 3, pp. 499–511, 2007.

[16] S. McCreary and K. Claffy, "Trends in wide area IP traffic patterns - A view from Ames Internet Exchange," in *ITC Specialist Seminar*, Monterey, CA, Sep 2000.