Lightweight benchmarking of platforms for network traffic processing

Pavol Korcek, Martin Zadnik
Brno University of Technology, IT4Innovations Centre of Excellence
Bozetechova 2, Brno, Czech Republic
Email: {ikorcek, izadnik}@fit.vutbr.cz

***Abstract*—Embedded processors seem to be a viable solution for network traffic processing. We can observe that the current network development boards utilize ARM, MIPS rather than specialized network processors. The processors for embedded applications are low cost, low power but their performance is not clear. In this work we aim at revealing their performance in terms of their throughput and processing power. To this end, we select three network processing functions such as longest prefix match, filtering and pattern matching. We benchmark several available platforms with embedded processors by implementing and running these tests in a controlled environment.**

I. Introduction

Building low cost specialized network devices is more and more challenging due to increasingly higher network bandwidth. Even more so, if the goal is to achieve also low power consumption which might allow the device to utilize Power over Ethernet or to withstand short-term power failures using batteries.Our intention is to find out whether currently available platforms, from the area of embedded networking, may address this challenge, i.e., enough performance and low power consumption.

There are several off-the-shelf platforms which might be customized to perform specific network traffic processing, e.g. home-based routers. Some of these routers support OpenWRT [1] which also provides API to program packet handling. Other platforms constitute of development boards and low power general purpose platforms with specialized toolchain.

Although the specification of these platforms is clear from the component perspective their performance in case of network traffic processing has not been well documented.

General metrics, such as frequency, provides only rough estimation of the performance. The suitability of a given platform for a particular application should be revealed through benchmarking.

Our goal is to design and implement a test suite and report on the results we obtained during measurement of several platforms which might serve for light-weight processing of network traffic such as intelligent taps, probes, tunnels, load balancers, etc.

II. Related work

Specialized benchmarks have been proposed for specialized processors such as benchmark for DSP processors by Berkeley Design Technology or MediaBench [2].

Also network processors (NPU) received their specific benchmarks. Commbench [3] and NetBench [4] benchmarks contain various tasks such as headerprocessing applications (Radix-Tree Routing, packet fragmentation, Deficit round robin scheduling, tcpdump kernel) and payload processing (ciphering, compressing, Reed-Solomon forward error correction, jpeg compression).

Further, the Embedded Microprocessor Benchmark Consortium [5] defines networking benchmark within its large set 34 application benchmarks. The EEMBC networking benchmarks consists of Patricia route look-up, OSPF Dijkstra's algorithm, packet duplication.

Our report contributes to and extends previous work in three aspects. We test processors for embedded (mostly networking) applications in the light of network traffic processing and we report the obtained results. Besides the processors performance, we also test network throughput of the complete platforms which host the processors. We propose new test suite which consists of specific state-of-the-art network algorithms.

III. Design of tests

We propose two types of tests. The first is focused on a measurement of network throughput during routing, forwarding or packet capture whereas the rest of the tests is focused on application-specific network operations.

A. Test of network throughput

Throughput tests aim at measuring throughput of a device when it is deployed in the network and processes passing traffic. In case of a device with a single network port, it is possible to measure throughput when the device captures or transmits packets. Whereas for multi-port devices the tests are designed to measure throughput the device is capable to achieve when forwarding or routing passing packets. The throughput is measured in a number of bits per second on several packet lengths (64, 256, 512, 1024, 1500). The throughput is measured using *iperf* tool [6]. In case of forwarding and routing throughput, the iperf client and server are hosted by high-performance PCs which are connected together via the device under test (DUT). In case of a single port device, the DUT is connected directly with the measurement PC and the DUT runs once as an iperf client and once as an iperf server.

978-1-4673-1188-5/12/$31.00 ©2012 IEEE

B. Performance measured on network algorithms

We select three basic network algorithms in order to measure DUTs performance when executing specific tasks upon an arrival of each packet. We consider the selected algorithms to be the most common in the domain of application-specific network traffic processing and measurement.

1) Test of LPM: Longest Prefix Match (LPM) is an operation essential for routing and IP packet classification. The operation leads to many memory accesses due to a trie structure. To address this potential bottleneck there are many variations utilizing optimized and extended tries and hash tables. Still the number of memory accesses might be an issue and therefore it is important to test promising LPM algorithms.

TreeBitmap (TBM) [7] is based on a concept of a Trie. It assembles multiple nodes of a trie into multinodes. Each of these nodes represents a full subtree of the same shape. A lookup operation upon such a structure allows traversing multiple tree-levels by a single memory access which reduces the number of total accesses per a complete lookup. Moreover, it provides quite efficient range encoding of the multinodes which reduces memory consumption.

Shape Shifting Tries [8]. Similarly to TBM, the SST modifies a simple trie. It assembles multiple nodes in a multinode with the main difference that the shape of each subtree represented by the multinode may be different to others and is stored in each multinode. This way a sparsely occupied tree may be represented more effectively which leads to fewer memory lookups but higher computational requirements.

The number of bits traversed at each level of the search through the tree is called the stride length. We experiment with two stride lengths for each look-up algorithm to see the impact of the stride on the performance.

2) Test of filtering: Effective filtering scheme is essential for applications which process only a portion of the whole traffic. The filter must be simple enough to deal with a full traffic rate and at the same time it has to consume only a small amount of resources. Generally, Bloom filters and their variants are considered very effective and have been proposed as a solution for many applications. We test basic Bloom filter and Counting Bloom filter. The test inherently includes computation of Bob Jenkin's hash function (lookup3) with various seeds. The hash values are used as indexes to access the filter.

Bloom Filter (BF). Bloom filter [9] is a probabilistic datastructure that captures information about the presence of a given element in a set which elements were previously fingerprinted into the datastructure. It provides a false answer in case when the element was not fingerprinted but its fingerprint overlaps with fingerprints of others. The underlying datastructure is a one bit-wide array. An element is fingerprinted into the array by setting up ones at positions indexed by multiple hash functions computed on the key of the inserted element. The query is performed by computing hashes on the queried element and inspecting bits at positions given by the hashes. If all bits are set up to one then the element has been inserted or it is a false positive with certain probability. Basic Bloom filter supports only insert and query operations. The removal of a fingerprint would invalidate other fingerprints as well.

Counting Bloom Filter (CBF). CBF [10] extends Bloom filter with the ability to delete an element from a filter. CBF underlying datastructure is an array of counters (suggested bit length of a counter in literature is 4 bits). The fingerprint of an element is inserted into the filter by incrementing counters at positions indexed by multiple hash functions. Whereas, by decrementing the counters the element is removed. If indexed counters are greater than zero then the element is present in the set. There are several parameters (of both Bloom filters) that need to be set such as number of elements in the inserted set, number of bits/counters of the array, probability of false positive. During testing these parameters are set up to different values but clearly the whole parameter space cannot be evaluated.

3) Test of Pattern matching: The tests above are focused on network or transport layer. But many applications require to parse application layer or to inspect packet payload. This constitutes of pattern or regular expression matching on a fairly large database of expressions. We test an algorithm based on deterministic automaton as well as an algorithm based on a combination of deterministic and non-deterministic automaton which promises better theoretical cost.

Delay Input DFA. D^2FA [11] is based on common deterministic finite automaton (DFA). It extends DFA with additional implicit transitions. Implicit transition is taken when no other transitions from the current state are allowed. The implicit transitions may reduce the total number of states necessary to represent the whole pattern database. The next state logic is encoded in a transition table. Upon each incoming character, one or more memory accesses are required into the transition table. The current state is saved in a register.

Hybrid FA (HFA). HFA [12] combines DFA and non-deterministic finite automaton (NFA). HFA utilizes NFA to represent regular expressions and DFA to represent patterns. Such division allows reducing the number of states hence reducing the state transition table. On the other hand, evaluation of NFA parts may require evaluating potentially viable transitions.

IV. Evaluation

The evaluation of selected embedded platforms was performed according to the proposed tests. The evaluation of raw throughput was followed by performance measurement of several network algorithms.

A. Tested platforms

It is usually not straight-forward to alter functionality of home-based routers. It requires replacing an original OS with an open-source OS such as OpenWRT. Needless to say, that the router must be ready to support such an operation. There might be obstacles such as insufficient quantity of flash or RAM, no Linux/driver support or too old processor.

Table I lists tested platforms with their configuration. First three selected platforms are based purely on the MIPS archi-

tecture. First one, Linksys WAG160N, is widely used home-based router but for 100 Mbps networks only. Next two devices are almost the same. The only difference between the second D-Link DIR-825 and the third Ubiquiti platform is in the size of the main memory.

Next platforms are based on well-known ARM architecture. First one, denoted as Econa is a small 1 Gbps development platform with quite an old type of processor running only at frequency of 250 MHz but with a quite large main memory (512 MB, compared to other selected platforms). Although we made algorithmic tests for this platform, we are not going to show them in this work. We made this decision due to very low speed caused not only by small processor frequency but also because of old processor architecture. On the other hand, this platform provides a unique hardware support for packet processing (HNAT - Hardware Network Address Table) accessible using special driver from applications. Whenever possible, the throughput tests on this platform explore available variants, i.e., the fast data path using HNAT as well as the slow path through processor. The hardware support can be disabled by not uploading its associated driver module. Second ARM-based platform is Avila GW2348 with Intels' XScale processor running at 533 MHz. This is a specially modified processor (DES, AES hardware encryption support) for network applications running on 100 Mbps networks. Last one in ARM-based group is a classical network disc - Seagate Dockstar. It is equipped with 1.2 GHz processor of a bit older ARM architecture compared to the previous one. This platform has only one 1 Gbps port.

MikroTIK RB800 is a router board used not only for small to medium enterprises networking. Its processor is based on Freescale PowerQUICC3, which is the third generation of processors especially designed for networking. Beside L1 cache, this processor has also 256 kB of L2 cache. It also utilizes a different type of OS (RouterOS, deployed by MikroTIK) which supports its hardware features correctly. On the other side, it was not possible to run algorithmic tests due to no utilizable toolchain support for this platform.

Finally, we select to test two personal computers with one 1 Gbps network port. The aim of this selection is to demonstrate a gap, if there is any, among embedded and common processors. First one, small computer eeePC is based on Intel Atom D510 running at 1660 MHz with 2 GB main memory and 1 MB L2 cache. The Intel Atom processor design aims at deployment in embedded computing [13]. The last one in Table I is an EsprimoPC equipped with fast processor. It is Intel Core2Duo running at 2.4 GHz with its 3 MB L2 cache and 3 GB main memory.

B. Throughput tests

The throughput measurement was performed by *iperf* tool as proposed in the design section. Besides various packet lengths we also tested several combinations of network and transport protocol, i.e., TCP/UDP for both IPv4 and IPv6, if IPv6 is available on the given platform.

We assume that tested embedded platforms achieve lower network throughput than current personal computers. Therefore we utilize ordinary PCs as packet generators or capture machines and measure the bottleneck of the DUT very cheaply. Whenever the throughput of any tested DUT is close to the reference PCs throughput it is not possible to estimate maximum throughput. Naturally, the first test was carried out to reveal the throughput of two directly connected PCs.The lowest throughput of 300 Mbps is achieved on the shortest packet-length. The full link capacity is reached at length 512 and higher.

Multi-port devices are compared in forwarding and routing performance whereas single port devices are evaluated in terms of their capturing (iperf client) and transmitting (iperf server) throughput. The methodology of throughput measurement is managed by iperf internally. In case of TCP iperf tunes TCP control parameters to achieve maximal bandwidth utilization with no packet loss and reports the achieved throughput. In case of UDP, we setup the bandwidth to 1 Gbps and observe reported packet loss. Only correctly delivered packets are accounted for the bandwidth report. Each test of particular configuration lasted 10 s and was repeated 5 times. The presented values in graphs are arithmetical average of these 5 measurements. We observed that the deviations of successive tests were at most 2% of the average results in all cases.

Forwarding. The achieved forwarding throughput for TCP and UDP is depicted in Fig. 1 and 2 respectively. If a platform supports IPv6 then the results are stated for both protocols and can be recognized by the suffix in the label. The Figures show that the highest forwarding throughput is achieved by D-Link and Ubiquiti consistently for TCP and UDP. Forwarding is the only case when the throughput of some DUTs is higher than the throughput of reference PCs as it is most likely performed by dedicated chip in tested platforms. From the platforms supporting 1 Gbps Ethernet, Econa achieves the lowest throughput when tested without enabled HNAT. If its HNAT is utilized then its forwarding throughput exceeds the reference PC as well. Therefore the worst forwarding throughput from measured 1 Gbps platforms is achieved by MicroTik. LinkSys as a 100 Mbps router almost achieves its maximum. In case of TCP, the forwarding throughput grows with increasing packet length in most tests whereas, in case of UDP, better forwarding bandwidth is achieved using 1024 B long packets. This happens due to a limited buffer size of the DUT when the throughput limit of the DUT is reached. The buffer is more likely to accommodate a shorter packet, moreover, if there is a packet loss then it is more efficient to lose shorter packet than longer one.

Routing. The achieved routing throughput for TCP and UDP is depicted in Fig. 3 and 4 respectively. For TCP, The Econa platform with enabled HNAT outperforms other platforms. The second best ends up Ubiquiti, surprisingly, when routing IPv6 traffic. D-Link achieves quite low throughput for both IPv4 and IPv6 traffic. Again, routing throughput of UDP reveals that it is better to use packets of length 1024 B to achieve better link utilization.

TABLE I: Configuration of tested platforms.

Label	Processor type	Architecture	Frequency [MHz]	Cache [kB]	Memory [MB]	OS type
Linksys WAG160N	Broadcom BCM6538	MIPS	300	I:32, D:16	32	OpenWRT
D-Link DIR-825	Atheros AR7161	MIPS	680	24	64	OpenWRT
Ubiquiti	Atheros AR7161	MIPS	680	24	128	OpenWRT
Econa	Cavium StarCS1102	ARMv4T	250	I:16, D:16	512	Linux, 2.6.16
Avila GW2348	Intel IPX425	ARMv5 XScale	533	I:32, D:32	64	OpenWRT
Seagate Dockstar	Marvell Kirkwood	ARMv5TE	1200	I:16, D:16	128	OpenWRT
MicroTIK RB800	Freescale MPC8544	Power QUICC3	800	I:32, D:32	256	RouterOS 4.0
eeePC	Intel AtomD510	-	1660	L2:1024	2048	Linux, 2.6.35
EsprimoPC	Intel Core2Duo	-	2400	L2:3072	3072	Linux, 2.6.38

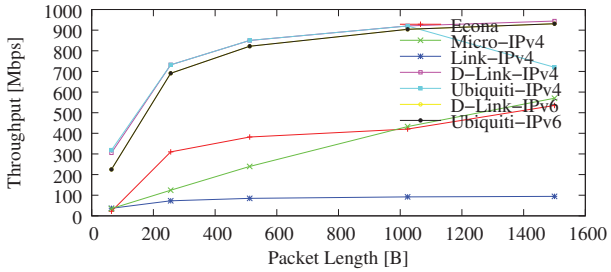

Fig. 1: Forwarding throughput TCP.

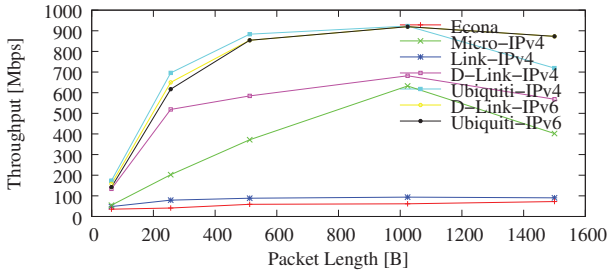

Fig. 2: Forwarding throughput UDP.

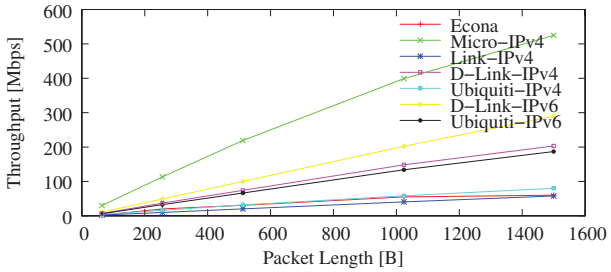

Fig. 3: Routing throughput over TCP/IPv4.

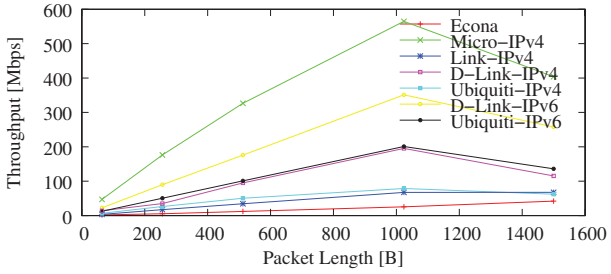

Fig. 4: Routing throughput over UDP/IPv4.

Iperf capture and transmit. The achieved throughput for TCP client and server configuration is depicted in Fig. 5. The Figure shows that the throughput of all devices with either IPv4 or IPv6 configuration is quite similar. Surprisingly, Seagate is far slower for UDP (350 Mbps for longest packets) than expected considering 1.2 GHz processor. All tested devices achieve better throughput when receiving packets. The difference is approximately 100 Mbps in comparison to transmitting throughput.

The achieved throughput for UDP client and server is not depicted due to a lack of space. In case of this measurement with iperf, the throughput of UDP is always lower than of a TCP, but remarkably, UDP/IPv6 traffic achieves better performance than UDP/IPv4.

During throughput tests, we have made several observations:

- Specialized HW support can significantly improve throughput performance
- UDP is likely to reach overall lower throughput with correctly delivered packet than TCP
- IPv6 routing might be on some platforms faster than IPv4
- Processor frequency influences the throughput less than expected

C. Algorithmic tests

The proposed tests of network algorithms were implemented in C. None of the algorithms utilizes any specific feature of the platform (e.g., there is only one instance of the algorithm running with no extra parallelism although dual core processor might be available). The tests are intentionally designed and implemented in such a way that they can be easily compiled

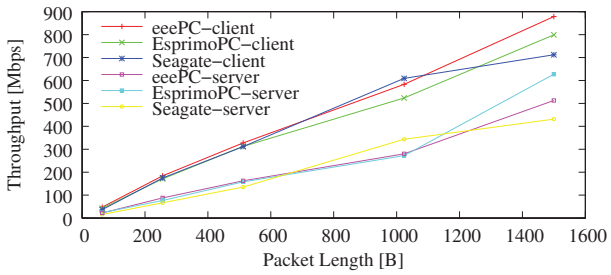

Fig. 5: Throughput of DUT running iperf receiving TCP.

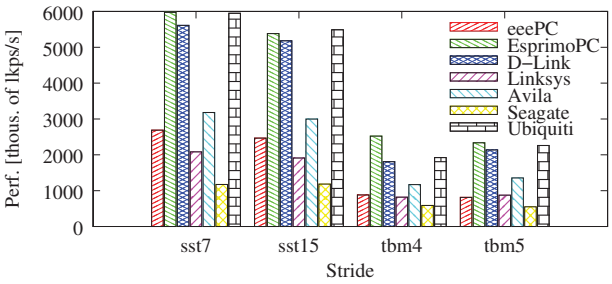

Fig. 6: Performance of LPM algorithms.

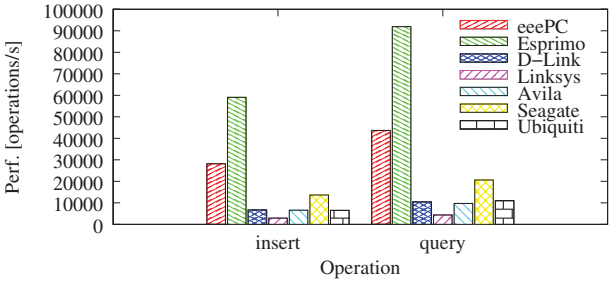

Fig. 7: Performance of filtering with Bloom Filter.

for any platform and any OS. This is achieved due to independence of the test suite on any specific software library and independence on any specific packet interface.

Rather than to read input data from a packet interface, the input data are retrieved directly from a memory where they are loaded from a file before a test starts. As a result the overhead of a packet interface is not accounted for. This overhead is measured separately during network throughput tests. The intention of algorithmic tests is to measure solely the fitness of a specific processor architecture (ALU, instruction set, CPU pipeline, memory hierarchy, etc.) for a given network operation. It takes various time each architecture to finish processing the input dataset. The metric we use for comparison is the number of operations per second. An operation consists of an access to the block of input data (block of data represents a packet or packet header), execution of a network algorithm and producing the result which is stored in the memory so that correctness of the output can be always verified. The number of operations per second gives an upper bound on what is possible to achieve on a given platform since in a real deployment the overhead of the packet processing must be accounted for.

We measured the performance with *gcc* compiler optimization set up to various levels. In this paper we report only the fastest results which were consistently achieved with optimization *-O3* on all platforms. The configurations and workloads of all algorithmic tests are summarized in Table II.

TABLE II: Workload and configuration for tests.

LPM	Prefixes [#]	Depth [av,max]	Queries [#]
BGP-FIT	1009	1.2,5	23204
BGP-BUT	10084	1.3,6	3382

Filt.	Size [items,bits]	Hashes [#]	Insert [#]	Query [#]	Remv [#]	Req. [#]
BF	64K,64K	8	192000	192000	-	-
CBF	64K,256K	8	192000	192000	96000	96000

Matching	Patterns [#]	Regexp [#]	D2FA States [#]	HFA States [#]
aut1	1	0	6	8
aut2	2	1	16	18
aut3	0	35	231	44

LPM. We use two BGP tables and we test TBM with stride length 4 (tbm4) and 5 (tbm5) and SST with stride length 7 (sst7) and 15 (sst15) for each table. We display results for the larger dataset only as the evaluation results are consistent on both tables.

Figure 6 compares performance of TBM and SST in the number of lookups per second for each platform. The interesting thing to notice is that D-Link and Ubiquiti (MIPS architecture) work very close to EsprimoPC (Intel architecture) which outperforms any embedded processor in other tests. Also, eeePC (another Intel architecture) is nearly two times slower compared to D-Link or Ubiquiti. EeePC is also slower than Avila (ARM architecture) running only at 533 MHz. Moreover eeePC performance in this task is very close to only 300 MHz Linksys (again MIPS architecture). It also seems that the embedded processors support SST better as the performance is constantly higher in comparison to TBM.

Filtering. Initially, all items in the set are inserted in the filter then the filter is queried with random lookups. In case of CBF, the query stage is followed by removal of $\frac{1}{3}$ of items from the filter. Finally, the resulting filter is queried again.Figures 7 and 8 show the performance measured individually for each operation.

We can see that each operation is approximately of the same cost across all platforms since all operations must compute hashes and access memory to alter the array. The exception is EsprimoPC in which the query operation runs very fast. We argue that such behavior is caused by large caches (3 MB) in comparison to other processor architectures. As a result the cache can accommodate whole filter. From within the set of classical embedded processors, the best performance is achieved this time by Seagate platform. In this task, all processors with MIPS architecture seem to be a bad fit.

Pattern Matching. Pattern matching performance was mea-

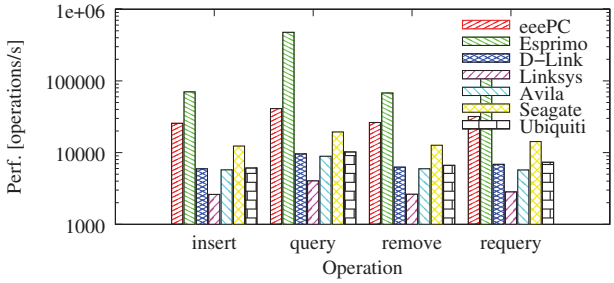

Fig. 8: Performance of filtering with Counting Bloom Filter.

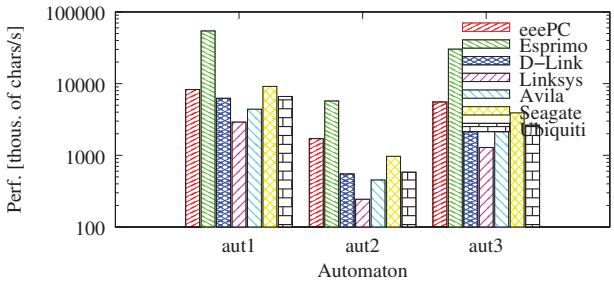

Fig. 9: Pattern matching performance of D^{2}FA.

sured using 3 pattern/regular sets, each with different characteristics. The results are presented in Fig 9 and 10. The measurements confirm the assumption that D^{2}FA is better for pattern matching while HFA performs better on the sets with mix of patterns and regular expressions.

Final note. We also have tested μ-Blaze processor (8.00.b) running at 50 MHz synthesized into Spartan-3 FPGA on the Xilinx Spartan-3E XC3S1600E development board. The performance is 10+ times lower than the performance of classical embedded processor which is in line with more than ten times lower frequency. We have evaluated two versions, one with branch predictor and two times bigger cache memories and the other without branch predictor. The branch prediction and increased cache size improved μ-Blaze performance by approx. 100% for LPM algorithms but only for ten percent in case of other algorithms.

During tests, we have made several observations:

- In case of filtering and pattern matching, the higher processor frequency the better result.
- For LPM, eeePC is more than two times slower than MIPS processor running at 680 MHz.
- Again for LPM, faster (1.2 GHz), but older (ARMv5TE) processor is slower compared to newer (ARMv5-XSCale) running on lower frequency (533 MHz).
- MIPS architecture is generally faster than ARM for LPM.
- Only a small change in μ-Blaze architecture invokes more than 2 times better performance.

V. Conclusion

The paper reported measurement results obtained on several currently available platforms for embedded networking. This paper also presented a design and implementation of a test suite to measure and compare performance of platforms for embedded network applications. The suite can be easily ported and compiled on any platform and OS. It consists of algorithm source codes, setup and measurement scripts, input data samples and generators. We provide all materials publicly available [1] so that designers and researchers can test and compare platforms of their interest. In our future work, we plan to evaluate further platforms when these become available.

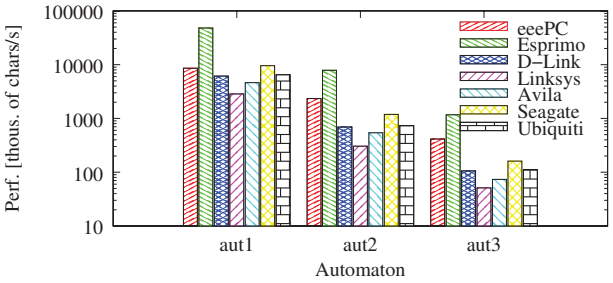

Fig. 10: Pattern matching performance of HFA.

Acknowledgments. This work has been partially supported by the Research Plan MSM 0021630528, IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070, the grant BUT FIT-S-12-1 and Sec6net project VG20102015022.

References

[1] Openwrt team. openwrt wireless freedom, 2012.
[2] C. Lee, M. Potkonjak, and W. H. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communicatons systems. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, MICRO 30, pages 330–335, 1997.
[3] T. Wolf and M. Franklin. Commbench-a telecommunications benchmark for network processors. In *Proceedings of the 2000 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 154–162, Washington, DC, USA, 2000. IEEE Computer Society.
[4] G. Memik, W. H. Mangione-Smith, and W. Hu. Netbench: a benchmarking suite for network processors. In *Proceedings of the 2001 ICCAD*.
[5] EEMBC. Embedded microprocessor benchmark consortium.
[6] Nlanr/dast. iperf tool, 2012.
[7] W. Eatherton, G. Varghese, and Z. Dittia. Tree bitmap: hardware/software ip lookups with incremental updates. *SIGCOMM Comput. Commun. Rev.*, 34:97–122, April 2004.
[8] H. Song, J. Turner, and J. Lockwood. Shape shifting tries for faster ip route lookup. In *Proceedings of the 13TH IEEE International Conference on Network Protocols*, pages 358–367, USA, 2005.
[9] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13:422–426, July 1970.
[10] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. An improved construction for counting bloom filters. In *Proceedings of the 14th conference on Annual European Symposium - Volume 14*, ESA'06, pages 684–695, London, UK, UK, 2006. Springer-Verlag.
[11] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner. Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In *Proceedings of the SIGCOMM'06*, pages 339–350, New York, NY, USA, 2006.
[12] M. Becchi and P. Crowley. A hybrid finite automaton for practical deep packet inspection. In *Proceedings of the 2007 ACM CoNEXT conference*, CoNEXT '07, pages 1:1–1:12, New York, NY, USA, 2007.
[13] Intel corp. intel atom processors n450, d410 and d510 for embedded computing - platform brief, 2010.

[1]The test suite is called procbench and can be downloaded from www.fit.vutbr.cz/research/view_product.php?id=174