2012 31st International Symposium on Reliable Distributed Systems

Data De-duplication and Event Processing for Security Applications on an Embedded Processor

Harsha Nagarajaiah, Shambhu Upadhyaya
Computer Science and Engineering
University at Buffalo
Buffalo, NY 14260 USA
{hpn, shambhu}@buffalo.edu

Vinodh Gopal
Intel Corporation
75 Reed Road
Hudson, MA 01749 USA
vinodh.gopal@intel.com

***Abstract*—Network security schemes generally deploy sensors and other network devices which generate huge volumes of data, overwhelming the underlying decision making algorithms. An example is corporate networks employing intrusion detection systems where there is a deluge of alert data, confounding the computations involved in sensor information fusion and alert correlation. One way to obtain fast and real-time responses is to preprocess such data to manageable sizes. In this paper, we show that data de-duplication using computationally efficient fingerprinting algorithms can provide real-time results. We present an algorithm which utilizes Rabin Fingerprinting/hashing scheme for the purpose of data de-duplication. We have implemented this algorithm on Intel Atom, which is a powerful, energy efficient embedded processor. Our study is intended to show that the relatively low performing embedded processors are capable of providing the needed computational support if they were to handle security functions in the field. When compared to the algorithmic performance on a high end system, viz. Intel Core 2 Duo processor, the positive results obtained make a case for using the Atom processor in networked applications employing mobile devices.**

Keywords-Alert correlation, Embedded processors, Fingerprinting, Mobile devices, Redundancy

I. INTRODUCTION

With the evolution of computer networks and the explosive growth of the Internet, information assets of both government and commercial organizations face credible threats from complex goal-oriented multistage attacks. Detection of these attacks as they unfold is essential especially in critical systems where failure to do so may be costly. This requires a high level understanding of the threat situation which is typically accomplished by event correlation and fusion of various cyber events [1]. In corporate networks such cyber events are generated by deploying intrusion detection and intrusion prevention (IDS/IPS) systems. Military systems, on the other hand, are more complex with airborne networks which are essentially three-dimensional mobile ad hoc networks [2] that generate a deluge of cyber events. Some of the processing nodes in these networks may have severe power constraints limiting their participation in alert correlation and fusion. Therefore, the traditional software-based approach to information fusion and situation awareness will have limited appeal in mission-critical applications where systems are subject to cyber attacks on a constant basis [3]. In order to make this solution applicable to real networks, especially when mobile/wireless devices are deployed, certain steps of alert correlation could be expedited by taking advantage of the architectural features of the processing nodes. In this regard, we propose to use Intel Atom processor owing to its flexible applicability to parallelization and flow processing [4] and its potential use in embedded applications.

One of the major computations in sensor information fusion and alert correlation is the removal of duplicate events for pattern matching or correlation. This is so because duplicate alerts may be generated when multiple sensors are deployed on the network to sense an event and hence network and host security systems may be presented with huge data from these sensors and other network sources. Typically, this data includes alerts streaming from sensors, firewalls, and anti-virus tools and system and network level data such as audit logs, packet information, netflows, protocol and port information. It is important to preprocess this data so that the correlation algorithms can be applied to a manageable data set to obtain real-time responses.

The main contribution of this paper is an algorithm which utilizes the Rabin Fingerprinting/hashing scheme [5] for the purpose of data de-duplication. We have implemented this algorithm on the Intel Atom processor, which is a powerful, energy efficient embedded processor. Our study shows that the relatively low performing embedded processors are capable of providing computational support if they need to handle security functions as well. When compared to the algorithmic performance on a high end system, viz. Intel Core 2 Duo processor, the positive results obtained make a case for using the Atom processor in networked security applications, especially in situations where a suite of mobile devices is employed.

The rest of the paper is organized as follows. In Section II, we give some background on Atom processor and Rabin Fingerprinting. The main algorithm for alert data de-duplication and processing is given in Section III. Details of the evaluation of our scheme using a repository of alert data from a federal project appear in Section IV. A discussion of the application domain for the proposed scheme is presented in Section V along with our concluding remarks.

II. BACKGROUND

The main experiment on data de-duplication using Rabin Fingerprinting is done on a first generation Intel Z5xx

1060-9857/12 $26.00 © 2012 IEEE
DOI 10.1109/SRDS.2012.18

(Atom) processor. The details of Atom and Rabin Fingerprinting are presented next.

A. Atom Processor

Intel® Atom processors [4] are power-optimized to deliver robust performance per watt, making them ideal for many embedded applications such as interactive kiosks, point-of-sale terminals, in-vehicle infotainment systems, media phones, industrial automation equipment, digital security systems, and residential gateways. These single-core processors are software-compatible with previous 32-bit Intel® architecture and complementary silicon. A three-chip solution is offered with the Intel Atom processor N270 and the mobile Intel® 945GSE Express Chipset.

Intel hafnium-based 45nm Hi-k metal gate silicon process technology reduces power consumption, increases switching speed, and significantly increases transistor density over previous 65nm process technology. Enhanced Intel SpeedStep® Technology reduces average system power consumption. Intel Hyper-Threading Technology (Intel HT Technology) available in designated Stock Keeping Units (SKU) provides high performance-per-watt efficiency in an in-order pipeline. HT Technology provides increased system responsiveness in multi-tasking environments. One execution core appears as two logical processors, and parallel threads are executed on a single core with shared resources.

The Atom processor is currently used in innovative in-vehicle infotainment (IVI) solutions. To keep pace with consumer demand, IVI developers and auto manufacturers need a platform that provides seamless integration between home, office and car, and bridges the gap from generation-to-generation of product development. Intel® architecture is highly interoperable with Wi-Fi, Bluetooth, cellular, WiMAX and emerging technologies like Ultra-Wideband, allowing OEMs to easily incorporate digital content into a head unit, now and in the future. Additionally, most consumer software is developed on and for the PC which lets developers easily add a breadth of applications to Intel®-based IVI systems via software-only upgrades. When coupled with extensive hardware and software from Intel's large community of developers, OEMs could benefit from rapid development and simplified upgrades at minimal cost. They include embedded lifecycle support, which protects system investments by enabling extended product availability for corporate networks that deploy intrusion detection and intrusion prevention (IDS/IPS) systems.

B. Rabin Fingerprinting

An n-bit message $m = m_0 \ldots m_{n-1}$, can be viewed as a polynomial f(x) of degree n-1 over the finite field GF(2) [6]. Let p(x) be a random irreducible polynomial of degree k over GF(2). The Rabin Fingerprint of message m is defined to be the remainder of division of f(x) by p(x) in GF(2) which can be viewed as a polynomial of degree k-1 or as a k-bit number [5]. It is fast and easy to implement, allows compounding, and comes with a mathematically precise analysis of the probability of collision. The probability of two messages r and s yielding the same w-bit fingerprint does not exceed $\max(|r|,|s|)/2^w$-1, where |r| denotes the length of r in bits. The algorithm requires the previous choice of a w-bit internal "key", and this guarantee holds as long as the messages r and s are chosen without the knowledge of the key.

Test results show that the time efficiency of Rabin's Fingerprinting method (linear in the length of the message) is comparable to other well known hashing functions while outperforming them in the sense of lower or even no collision occurrences [7]. Rabin fingerprints offer provably strong probabilistic guarantees that two different messages will not have the same fingerprint. Other checksum algorithms, such as MD5 and SHA, do not offer such provable guarantees, and are also more expensive to compute the fingerprints.

Data de-duplication is essentially a string matching problem using digital fingerprints. Randomly chosen irreducible polynomials are used to "fingerprint" bit strings or messages. This method is applied to produce a very simple real-time string matching algorithm [7]. The result of application of Rabin's method is shown to be independent of the choice of irreducible polynomials. The probability of error (different strings having the same fingerprint) decreases with the increase of the degree of the irreducible polynomial used. The performance of the method improves with the increase of the degree of the irreducible polynomial used and it is shown that with 64-bit fingerprint, the percentage of collision is 0 [7]. Due to this efficiency, we propose to use Rabin Fingerprinting for data de-duplication. Some other applications of Rabin fingerprinting are in detecting worms [8], [9], web cache [10], large file finding [11] and redundancy elimination in large collections of files [12].

III. DATA DE-DUPLICATION ALGORITHM

We have developed a data de-duplication scheme that is customized to remove redundant cyber events/alerts in a network with deployed IDS/IPS sensors such as SNORT. The algorithm is shown in Fig. 1 and described below in a step-by-step manner.

a. Input alerts (alert log files) from various alert sources are collated into a single input file. The alerts in this file are sorted by their respective timestamp values.
b. Based on a pre-specified threshold value of time, the alerts from the input file are split into multiple files. A single file contains alerts (with timestamp field removed) for the specified time duration. Each file forms an input to the Fingerprinting algorithm. Removal of timestamp leads to some degree of redundancy among the alerts (done to guarantee that there is sufficient redundancy in the data set for illustration purposes).

c. The Fingerprinting application processes each file and the resulting unique alerts, in the time duration under consideration, are stored in corresponding data structures. Internally the application uses Rabin Fingerprinting for fast pattern matching and data de-duplication. After redundant alert removal, the resulting unique alerts are stored in an appropriate data structure (such as the Multimap or Multihash [13]).

d. The output of each data structure which consists of unique alerts for the specified time durations are written to a single output file. This file represents a manageable data set for real-time processing, and serves as an input to the alert correlation algorithm (discussed elsewhere).

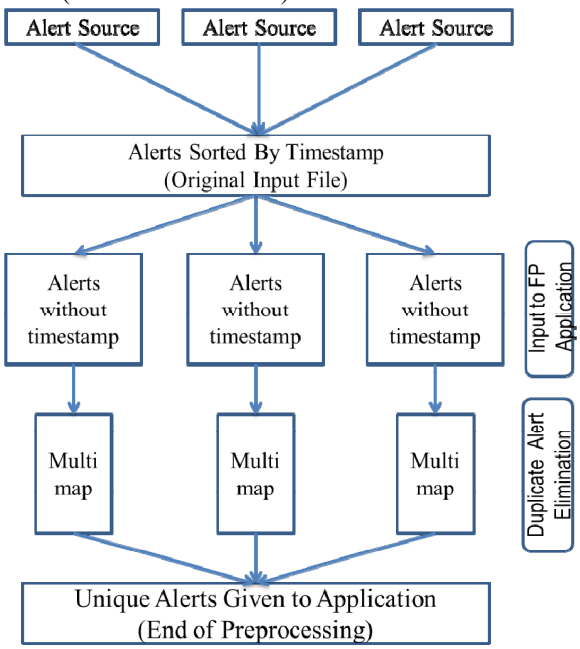

Fig. 1. The flow diagram of data de-duplication.

IV. EVALUATION

We have implemented the Rabin Fingerprinting (FP) based data de-duplication algorithm in our lab using Intel® Atom™ processor N270 and Intel® System Controller Hub US15W development kit. The development kit is shown in Fig. 2.

The algorithm is implemented in C++ and the code is executed on different machines/platforms. We have conducted controlled experiments using specific data sets. The data set for the controlled experiment is created using SNORT security alerts obtained from a federal project testbed [14] and Perl scripts. The timing analysis is performed on the processing unit of the Fingerprint application. The steps involved are – reading from alert files, duplicate alert elimination, storing unique alerts in an appropriate data structure and writing the output of all the data structures into a common output file.

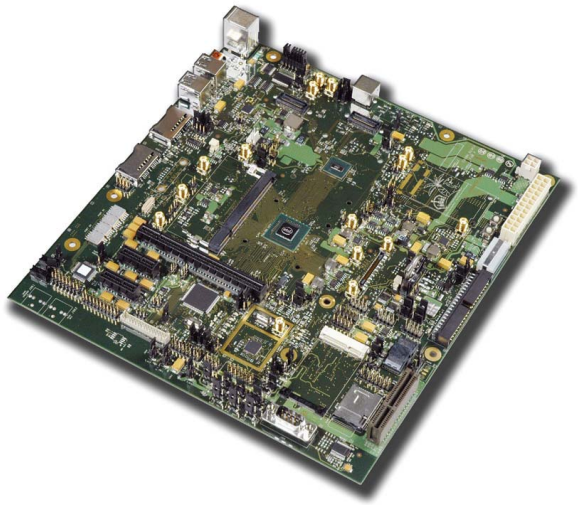

Fig. 2. Intel® system controller US15W development kit.

We first processed the alerts in the data set using UNIX sort command to determine the ground truth on the uniqueness of the alerts. The results from the Fingerprinting application were compared and found to match the ground truth. This validates the Fingerprint application's processing component. We also performed a runtime comparison of our fingerprinting based de-duplication with the results obtained using the UNIX "sort –u" command implementation. For simple and short alerts we observe that the sort approach closely matches the performance of FP. This is because the alerts considered here corresponded to a single line in the input file. If the input alerts span over multiple lines then the simple sort approach suffers from considerable overhead involved in converting to the necessary format both before and after processing the alerts. If we consider a complex alert data set where the individual alert spans over multiple lines (typical of SNORT alerts), the results show that the performance of the Fingerprinting application is significantly better. Fig. 3 illustrates that Rabin Fingerprinting based algorithm performs 10 times faster than the de-duplication using simple sort, which justifies the usage of Rabin Fingerprinting.

There are two versions of the FP code, viz. single threaded version and a corresponding multi-threaded version(s). There are different approaches to parallelize code execution. Some of the approaches are explicit threading, OpenMP, Intel Threading Building Block (TBB), Auto Parallelization, etc. In order to apply parallelization to the existing single-threaded version of FP code, we have considered using Intel TBB [15]. Intel TBB is used to express parallelism in the FP code. Intel TBB is not just a threads-replacement library but also represents a higher-level, task-based parallelism that abstracts platform details and threading mechanisms for scalability and performance.

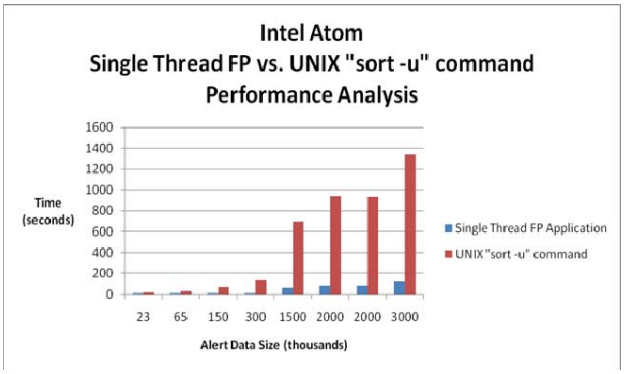

Fig. 3. Runtime performance of de-duplication algorithms.

In order to make the FP code multi-threaded, sections of the serial code suitable for parallelization are identified. The identified sections are in turn replaced by the corresponding parallel version. There are two versions of multi-threaded code for the same FP application. The two versions differ in the extent to which parallelization is applied. Version 1 has a lesser degree of parallelization compared to Version 2 of the multi-threaded code. After parallelization, performance comparison based on timing analysis is done in a manner similar to the single-threaded timing analysis. Fig. 4 illustrates the comparative performances of de-duplication with single thread and multi-threads. The performance gain due to multi-threading is approximately 1.2 times that of single threading. The gain by multi-threading is not so significant due to limited room for the parallelization of the FP algorithm. However, the data de-duplication itself is very fast. For example, 65,000 alerts are processed and redundancy removed under 2.5 seconds.

We wanted to see how the Atom results would stack up against those obtained with more powerful desktop CPUs such as the Intel Pentium Core 2 Duo processor. The details of the two platforms are as follows.

Pentium 4 Core 2 Duo Configuration is as follows: Freq: 2.66 GHz; RAM: 4 GB; Cores: 2 L2 Cache: 4 MB. Hyper threading Enabled: NO/YES.

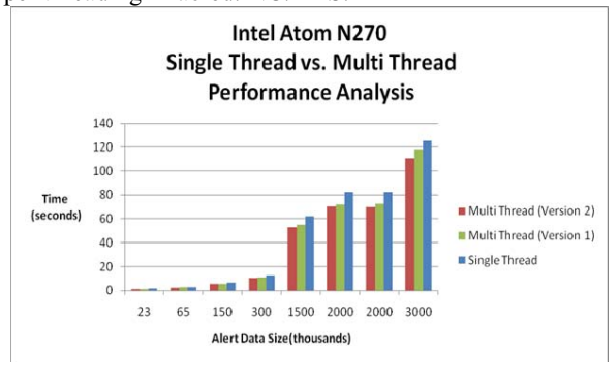

Fig. 4. Code execution time for single thread vs. multi-threads on Atom.

Intel Atom Configuration is as follows: Freq: 1.6 GHz; RAM: 1 GB; L1 Instruction Cache - 32 KB L1 Data Cache - 24 KB L2 Cache - 512 KB. Hyper threading Enabled: NO/YES.

Fig. 5 shows the relative performances on Atom and Pentium 4 (P4) with single threading. It can be seen that the P4 single threaded version executes 3.3 times faster than the Atom single threaded version. Considering that the Atom processor runs at much lower frequency than P4 and with RAM size one-fourth, the slowdown is only about 3 times. We have done experiments with multi-threaded versions 1 and 2 on Atom and compared the performance with multi-threaded versions 1 and 2 on P4. The slowdown is in the range of 3.5 to 3.9 times.

For completeness, the Rabin Fingerprinting scheme is contrasted with one other fingerprinting algorithm, also used for data de-duplication. We have considered the widely used Adler32 algorithm [16] for comparison. The performances of both multi-threaded and single threaded versions on Atom and Pentium Core 2 Duo processors are compared and a summary of results is given here (without any charts).

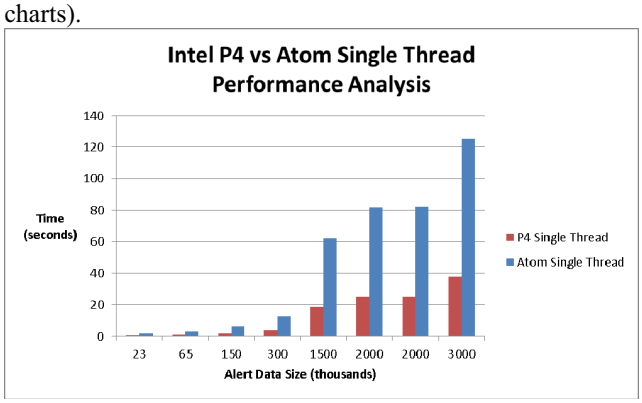

Fig. 5. Comparison of code execution times on Atom and Pentium 4 (single threaded).

On the Atom processor, the single threaded version of the Adler32 algorithm executes 1.7-1.8 times faster than the single threaded version of Rabin FP algorithm. The multi-threaded version of the Adler32 algorithm executes 1.5 times faster than the multi-threaded version of Rabin FP algorithm. On the Intel Core 2 CPU, the single threaded version of the Adler32 algorithm executes 1.5-1.6 times faster than the single threaded version of Rabin FP algorithm. The multi-threaded version of the Adler32 algorithm executes 1.3 times faster than the multi-threaded version of Rabin FP algorithm.

Though the Adler32 performs better (faster) than the Rabin FP algorithm, the accuracy of data de-duplication is not 100% in the case of Adler32. When the Adler32 algorithm generates the same checksum for two different alerts, these alerts are treated as though they are identical (collision) and it results in unique alerts being discarded. This may result in important alerts being discarded prior to being input to alert correlation engines. In the chosen alert

data set of 5,000 unique alerts, 14 unique alerts were not detected. In a set of 10,000 unique alerts, 24 unique alerts were not detected and in a set of 13,000 unique alerts, 33 unique alerts were not detected. In the case of Rabin FP algorithm, all the unique alerts are correctly determined as unique. Thus, Adler32 could prove to be detrimental if used for redundancy removal in real-time safety applications unlike the Rabin FP algorithm.

V. DISCUSSIONS AND CONCLUSION

Based on the results of the timing analysis, we can conclude that the Rabin Fingerprinting scheme used for data de-duplication is both feasible and efficient. We have shown that there is a significant performance gain in the execution of the Fingerprinting application on the Atom processor due to its inherent parallelization capability (i.e., hyper-threading). Also, the fingerprinting application performs very efficiently on common high end systems indicating that it could be deployed on a larger scale, e.g., corporate networks that deploy intrusion detection/prevention systems.

Our research has potential for significant impact on mobile application security such as the vehicular networks (VANET) security [17]. Second generation Atom processors are breaking into the In-Vehicle Infotainment (IVI) market [18]. In the case of security schemes deployed in VANETs, a vehicle may receive the same messages/alerts from different sources multiple times. This could be a result of message forwarding, repeated transmission of messages, and so on. This could lead to numerous duplicate alerts being provided as input to safety applications making real-time processing difficult. If we utilize data de-duplication algorithms in such scenarios, it will result in a manageable data set being provided as input to real-time applications. The duplicate alerts are filtered by the algorithm and hence the application processes useful unique alerts to provide real-time responses.

Also, in safety applications, Denial of Service (DoS) is one form of attack where the adversary can try to prevent access to a service or create false alarms about a non-existent situation by flooding the target system with repeated messages. The data de-duplication algorithm would discard such duplicate messages. The data de-duplication algorithm can also keep track of the number of discarded duplicate messages. If all the duplicate alerts are genuine, then the number of duplicate alerts helps the application to deduce the seriousness of the threat.

In VANETs, the data set size is small as compared to the large data set size in corporate networks. However, the fingerprinting application is effective for both small and large data sets. The Atom processor performs efficiently in both scenarios. Our research ensures that the Atom (embedded) processor can be used in security applications at virtually no extra cost, because significant performance gain is obtained by utilizing the inherent architectural capabilities of the processor. They can be especially useful in situations where mobile/wireless devices are deployed.

It is desirable to run our experiments on other generic processors as well as other embedded processors in order to generalize our results. This is part of our future work.

ACKNOWLEDGEMENT

This research has been supported in part by a grant from Intel Corporation and a grant from the Department of Defense (Grant No. H98230-11-1-0463). Usual disclaimers apply.

REFERENCES

[1] S. Mathew, S. Upadhyaya, M. Sudit and A. Stotz, "Situation Awareness of Multistage Cyber Attacks by Semantic Event Fusion", *IEEE MILCOM*, Oct. 2010.

[2] B. Ames, "Airborne Networking Challenges", *Military and Aerospace Electronics Magazine*, 2004.

[3] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz, "A Data Mining Analysis of RTID Alarms", *Computer Networks*, 34:571-577, 2000.

[4] "Intel® Atom™ Processor Z520, *Intel*, 2008, http://ark.intel.com/products/35466/Intel-Atom-Processor-Z520-%28512K-Cache-1_33-GHz-533-MHz-FSB%29

[5] M.O. Rabin, "Fingerprinting by Random Polynomials", *Center for Research in Computing Technology, Harvard University*, Tech Report TR-CSE-03-01, 1981.

[6] W.W. Peterson, *Error-Correcting Codes*, MIT Press, 1961.

[7] C. Chen and H. Lu, "Fingerprinting Using Polynomial (Rabin's Method)", *Term Project, U. of Alberta,* 2001.

[8] H. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection", *Proceedings of the 13th Usenix Security Symposium*, Aug. 2004.

[9] S. Singh, C. Estan, G. Varghese and S. Savage, "Automated Worm Fingerprinting", *Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation (OSDI),* 2004.

[10] L. Fan, P. Cao, J. Almeida and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol", *ACM Transactions on Networking*, vol. 8, No. 3, 2000.

[11] U. Manber, "Finding Similar Files in a Large File System", *USENIX Winter Technical Conference*, 1994.

[12] P. Kulkarni, F. Douglis, J. La Voie and J.M. Tracey, "Redundancy Elimination within Large Collections of Files", *Proceedings of the 2004 Usenix Annual technical Conference*, June 2004.

[13] Standard Template Library Programmer's Guide, Silicon Graphics International.

[14] R. Stapleton-Gray and S. Gorton, "Rendering the Elephant: Characterizing Sensitive Networks for an Uncleared Audience", *Proceedings of the IEEE International Information Assurance Workshop*, pages 208-214, West Point, NY, USA, 2006.

[15] Intel Threading Building Blocks (TBB) for Open Source. http://threadingbuildingblocks.org/

[16] P. Deutsch and J.-L. Gailly, *ZLIB Compressed Data Format Specification Version 3.3*, IETF RFC 1950, May 1996.

[17] P. Papadimitratos, L. Buttyan, T. Holczer, E. Schoch, J. Freudiger, M. Raya, Z. Ma, F. Kargl, A. Kung and J.-P. Hubaux, "Secure vehicular communications: design and architecture", *IEEE Communications Magazine*, vol. 46, no. 11, pp. 100-109, November 2008.

[18] Setting the pace in automotive technology, http://www.intel.com/content/www/us/en/embedded-developers-engineers/automotive-overview.html