Effnet ROHC™ (Robust Header Compression) Performance on Intel® Core™ Microarchitecture-Based Processors

Introduction

Effnet is a leading supplier of header compression software solutions to the network infrastructure and cell phone industry. They have offered software for high-performance network infrastructure products based on various types of processors, including Intel® Pentium® 4 processors based on the Intel NetBurst® microarchitecture for some time, and have recently validated products on Intel® Core™ microarchitecture-based processors, including the new Dual-Core Intel® Xeon® processors LV ATCA 5138$^{\Delta}$ used in the Intel NetStructure® MPCBL0050 Single Board Computer. These processors contain two separate execution "cores" on a single die and perform well on desktop, gaming, and server benchmarks. Robust Header Compression (ROHC) application workloads are more typical of "user plane" (often referred to as "data plane") communications workloads and are very different from desktop or server workloads. Utilizing the Effnet ROHC™ implementation designed to run on network infrastructure equipment, rather than the light-weight version designed to run on mobile devices such as cell phones, Effnet measured how well the multi-thread capable application would perform on the new Intel® multi-core processors.

Effnet first compared the performance of the Effnet ROHC solution on a core-for-core basis across two Intel® processors based on different microarchitectures; in other words a single core was used from each processor. An analysis was then performed to determine whether the Effnet ROHC software would scale to take advantage of all four cores available on the Intel NetStructure MPCBL0050 Single Board Computer, which has two Dual-Core Intel Xeon processors LV ATCA 5138.

This paper describes ROHC, the process Effnet used to test the Effnet ROHC software on Intel processors, and presents the test results.

Table of Contents

Overview of Robust Header Compression

The ROHC standard, known as RFC 3095, was developed in 2001 by the IETF. This compression scheme, which includes a compression algorithm and an information exchange protocol, can compress IP/UDP/RTP headers to just over one byte and is robust even in the presence of severe channel degradation. It can also compress IP/UDP and IP/ESP packet flows.

Header compression is used in many types of networks, including satellite communications networks, low-bandwidth wired networks, and private wireless networks such as Terrestrial Trunked Radio (TETRA). The widely deployed Frame Relay voice/data encapsulation protocol, typically used to bridge between local area and other local-area or wide-area networks, also utilizes header compression.

In addition to the information above, ROHC is also used in wireless network infrastructure equipment and mobile terminals (cell phones) to decrease header overhead, reduce packet loss, improve interactive response times, and increase Quality of Service (QoS) over error-prone wireless radio links. ROHC has been adapted to work with link layer characteristics like those used in GSM and CDMA networks, and is known as Link Layer Assisted-ROHC (ROHC-LLA).

The 3rd Generation Partnership Project (3GPP) standards group for the GSM/UMTS (W-CDMA) type of cellular network details IPHC and ROHC standards since Release 4. These schemes are essential for the successful deployment of equipment based on the Release 5 and 6 specifications, which introduce IPv6 and IP Multimedia Subsystem (IMS). The 3GPP2 standards group for the CDMA type of cellular network also specifies various QoS techniques including ROHC. The CDMA2000 EV-DO Rev A specification, for example, uses header compression to help enable operators to move from circuit-switched voice network infrastructure over to Voice over IP (VoIP). Figure 1 shows that header compression is implemented in handsets and all of the wireless infrastructure equipment upstream from the BTS/NodeB.

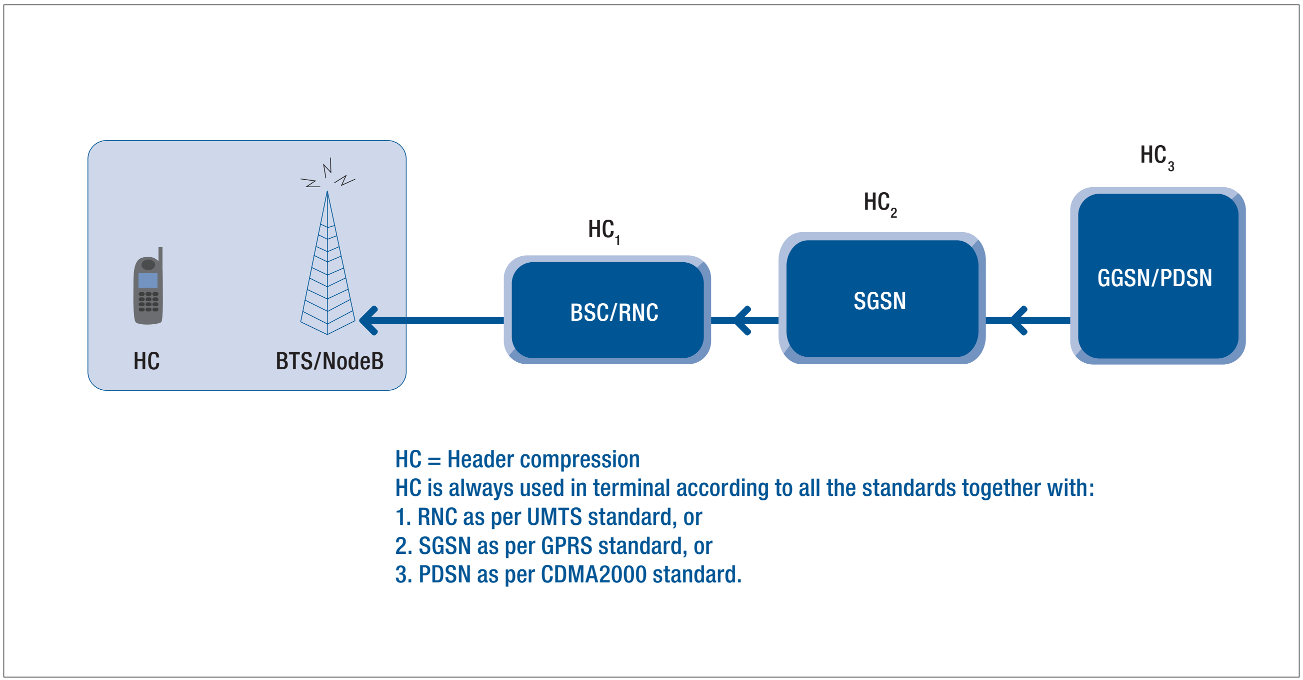
HC
BTS/NodeB
HC$_1$
BSC/RNC
HC$_2$
SGSN
HC$_3$
GGSN/PDSN
HC = Header compression
HC is always used in terminal according to all the standards together with:
1. RNC as per UMTS standard, or
2. SGSN as per GPRS standard, or
3. PDSN as per CDMA2000 standard.

Figure 1: Header compression in 3G wireless infrastructure equipment

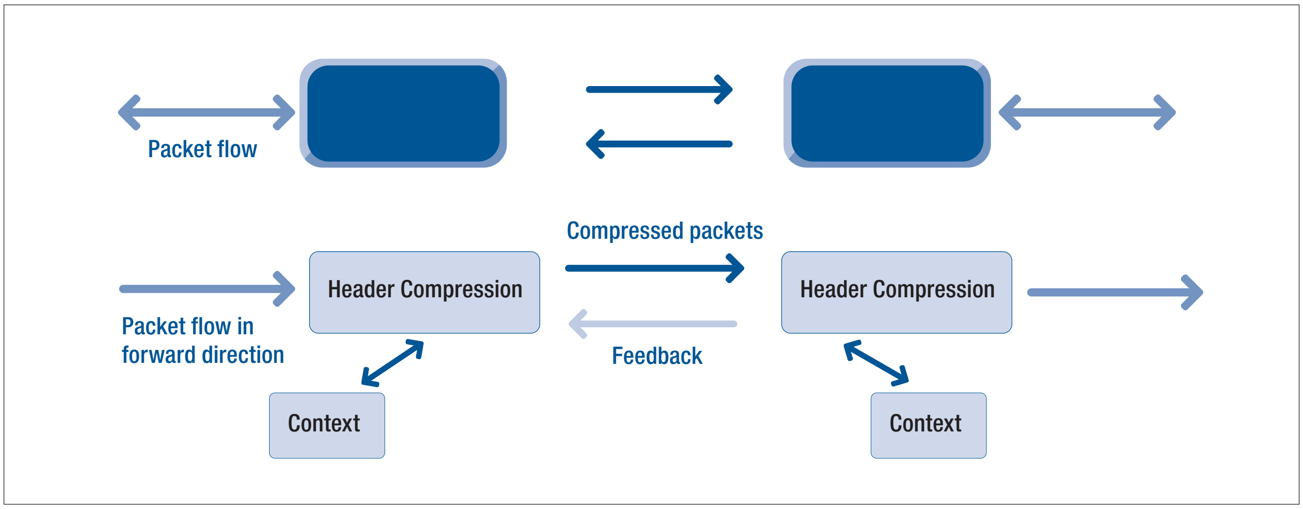

Figure 2: Header compression functional blocks

How ROHC Works[1]

Header compression is the process of reducing protocol header overhead in order to improve link efficiency while maintaining the end-to-end transparency.

Figure 2 shows the major functional blocks involved in header compression. Flow context is a set of values of packet header fields and the patterns of changes in those fields over time. This context is formed during the compression and decompression side for each packet flow. The first few packets of a newly identified flow are sent uncompressed so that a context can be determined on both sides. The number of these first few packets will vary depending on the Bit Error Rate (BER) and Round-Trip Time (RTT) of the specific link. Once the context is established on both sides, the compressor compresses the packets as much as possible until the flow contexts change or flow error rates start to increase.

Figure 3, on the next page, illustrates how a series of digits (in this case, the years 1800 through 1899) can be compressed by 72% using an encoding mechanism known as Window-based Least Significant Bit encoding (W-LSB). If the "1800" context is first established between the compressor and the decompressor, and the decompressor knows that only the final digit will vary, the compressor only needs to send the final nine digits to enable the decompressor to reconstruct the ten numbers in the first sequence. Then the context "1810" was established, and again only the next nine digits would need to be sent. This process was repeated so that 112 bytes of data were sent, as opposed to 400 bytes if compression had not been employed. The W-LSB algorithm together with the feedback mechanism makes ROHC very robust against bit errors on the link and also against errors introduced due to long RTTs. In addition, it results in very high compression ratios, which increase effective link bandwidth and decrease packet processing requirements in power sensitive downstream devices such as cell phones.

Figure 4, on the next page, shows the flow of packets over a hypothetical link. Once the context is established, ROHC sends packets with maximum compression. If the field pattern changes (as shown in the diagram by a silence period when neither party is talking) or if error conditions such as bit errors or packet loss occur, the context will need to be updated. Therefore, more information will need to be sent to the decompressor using slightly larger, but still compressed, packets. Once the context is updated, the link reverts to fully compressed packets. The feedback mechanism from the decompressor back to the compressor, which triggers this type of behavior, is one of the reasons why ROHC is so robust.

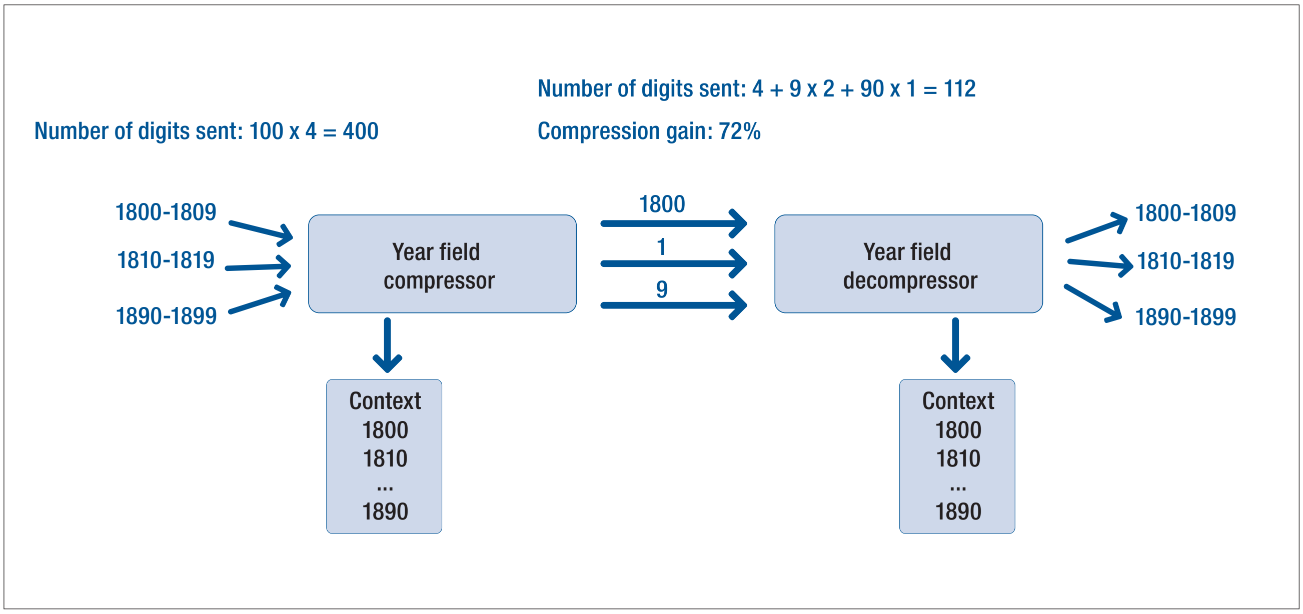
Number of digits sent: 100 x 4 = 400
Number of digits sent: 4 + 9 x 2 + 90 x 1 = 112
Compression gain: 72%
1800-1809
1810-1819
1890-1899
Year field compressor
1800
1
9
Year field decompressor
1800-1809
1810-1819
1890-1899
Context 1800 1810 ... 1890
Context 1800 1810 ... 1890

Figure 3: An example of Window-based Least Significant Bit (W-LSB) encoding

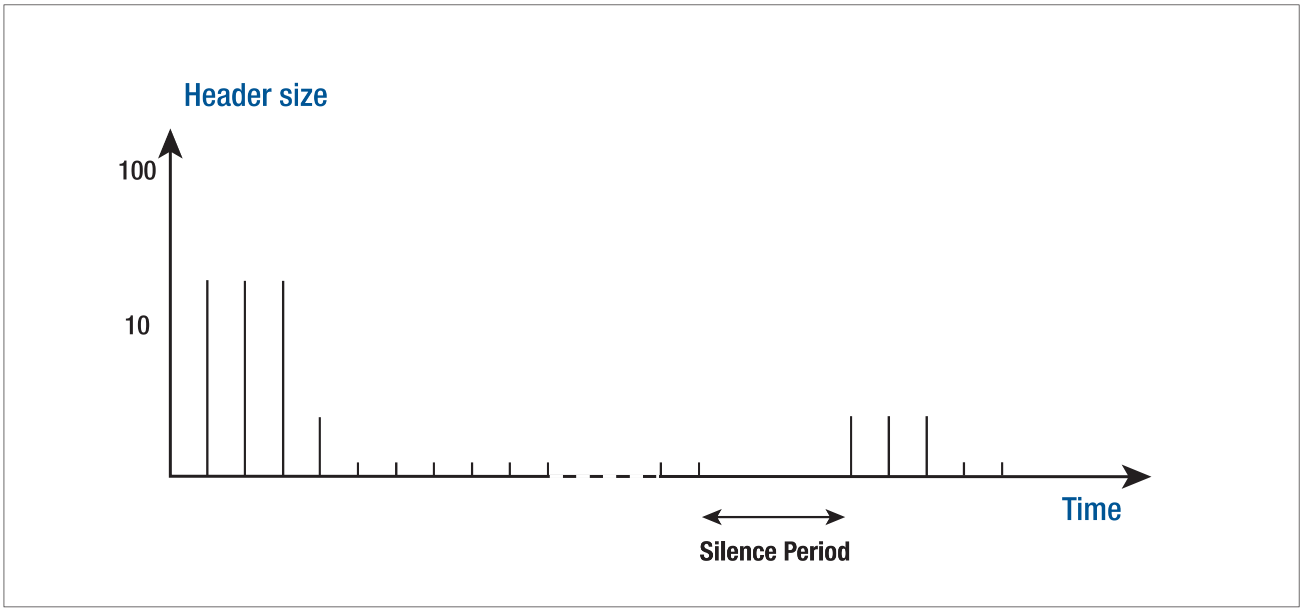

Figure 4: Packet-size variation during VoIP flows

Table 1: Header compression gains

Protocol Headers	Total header size (bytes)	Min. compressed header size (bytes)	Compresson gain (%)
IP4/UDP	28	1	96.4
IP4/UDP/RTP	40	1	97.5
IP6/UDP	48	3	93.75
IP6/UDP/RTP	60	3	95

ROHC Benefits

ROHC can achieve compression ratios well over 90% for the most common link, network and transport layer protocols as shown in Table 1.

Table 2 shows the advantages of using ROHC in wireless networks. The data in Table 2 is derived from a test scenario based on a VoIP packet flow using a header chain of IPv6 (40 bytes) and UDP/RTP (total 20 bytes), and carrying one frame of an AMR-NB codec at 12.2 kbps (31 bytes). Such a flow was sent every 20 milliseconds (ms) over a simulated wireless link that used an uncorrelated BER model with a BER level of 10-3. The simulation ran for 120 seconds and compressed 6000 packets. As Table 2 shows, ROHC reduced the total number of packets lost from 52% to 24%. It also decreased the required call bandwidth by 63%, header sizes by 94%, and the number of packets lost due to the header field errors from 38% to 3%.

Table 2: Advantages of ROHC

	Without ROHC	With ROHC
Total Packets Transmitted	6000	6000
Packets Lost	3125 (52%)	1448 (24%)
Call Bandwidth (Kbps)	35.5	12.9
Average Header Size	60	3.1
Packets Lost Due to Error in Header	2309 (38%)	188 (3%)

The benefits of ROHC are most apparent in CODECs that can handle bit errors in their payload as demonstrated in Table 2. Even without using such a CODEC, ROHC can reduce packet loss by approximately 50%. On certain wireless links, the compressed packets can be sent in just one link frame which is a highly efficient use of radio resources. These ROHC characteristics make it possible to incorporate high-quality VoIP services in wireless networks.

Hardware Resource Requirements

As shown, header compression greatly reduces the amount of data that must be sent and increases the effective link bandwidth. This is extremely important because according to RFC 3095, “Bandwidth is the most costly resource in cellular links. Processing power is very cheap in comparison. Implementation or computational simplicity of a header compression scheme is therefore of less importance than its compression ratio and robustness.”[2]

ROHC uses various encoding schemes including W-LSB and other specific field pattern-based encoding algorithms. CRC is used in most or all of the packets conveying context information to protect against residual errors and to maximize context validity. Packet loss and reordering, the most common error events on wireless networks, are handled by the compression/decompression method outlined previously.

This means that ROHC must actually save a larger amount of context information than any single non-compressed header would contain. Because the context information is saved, fewer processor cycles are needed to process a given packet. The processor’s L2/L3 cache also helps limit the total number of processor cycles required since accessing cached data is more efficient than accessing data from system memory. But the trade-off for reduced processor cycles is that the system will require larger amounts of memory and higher memory access bandwidth to support the larger amount of context information.

In addition to saving context, a complete header compression solution also includes packet classification and context management modules. Flow classification can be based on hash-table algorithms which use hash-keys to identify context information. These algorithms are also processor intensive and also require suitable memory bandwidth so that the identified context can be accessed as quickly as possible.

Therefore, a balance must be achieved between minimizing processing cycles and also keeping memory sub-system requirements within bounds. This can be achieved by using sophisticated methods to control context memory size. The goal is to keep memory size, bandwidth and processor cycles – required for processing and memory access – within the limits required to attain a specific processor load factor at a high number of flows.

As noted in the introduction, ROHC workloads are very different from typical desktop or server workloads. Unlike graphics or video-processing, for example, ROHC does not use any floating-point operations. Most of the processing involves branch (not loop) processing using logical and integer operations. Maintaining state after each compression and decompression operation is essential, so memory access (read, write, or both) happens frequently, but in a highly random fashion, with few if any sequential memory accesses. The most suitable hardware platform for ROHC is therefore one in which high processor performance is combined with a large L2 cache, where memory bandwidth is high, and where branch prediction is highly optimized.

The previous discussion is focused primarily on network infrastructure equipment with high-end hardware architectures. A good ROHC implementation for user equipment handsets has only to support a very small subset of contexts, so it is highly optimized and configured to limit the number of processor cycles it requires, and to use the memory resources extremely efficiently.

ROHC Performance on Intel® Architecture Processors

On cell phones, ROHC performance is typically measured as processor cycles per packet. On network infrastructure equipment, however, processor load per number of compressed/decompressed VoIP flow pairs is the more meaningful measurement. Therefore, the Effnet test application, ROHC Load Generator was designed to generate and measure VoIP flow pairs. The application was compiled on both the Intel® C++ Compiler and the standard GNU* C Compiler for Linux,* with various optimization flags, and tests were run on each set of object code.

The Effnet ROHC™ version 2.2 embedded in ROHC Load Generator was first run on a Dell Dimension* 8400 desktop, based on the Intel Pentium 4 processor, in order to provide baseline performance measurements for a processor based on Intel NetBurst microarchitecture. The same performance measurements were then done on the Intel NetStructure MCPBL0050, which uses the Dual-Core Intel Xeon processor LV 5138, based on the Intel Core microarchitecture.

Test Methodology

The goal of the test was to determine how many VoIP flows could be processed by the Effnet ROHC software for a range of processor loadings during the 20 millisecond (ms) signaling interval used between consecutive packets in a typical VoIP flow. The ROHC Load Generator software sets a 20 ms system timer, generates packets for all flows, and then hands them over one by one to Effnet ROHC for processing. The test application can vary the number of flows generated and processed and the various processor utilization rates can be measured using the Linux **/proc/stat** command.

The ROHC Load Generator software runs in Linux user space. A typical packet is loaded into memory and then altered as required for each flow before being handed off to Effnet ROHC. This process is extremely efficient and its effect on processor utilization is negligible. As noted previously, the ROHC Load Generator is single threaded. So for the tests involving multiple processor cores, a single copy of the load generation software and the ROHC Load Generator application was loaded onto each individual core with the Linux system call **sched_setaffinity**.

The following test methodology assumptions and conditions should be noted:

- There was no packet routing or reading from/writing to the network interface, so time to generate packets can be ignored. This approach was taken in order to measure the efficiency and performance of the Effnet ROHC software by itself.
- Effnet ROHC processes each flow via classification, compression and decompression. The test simulation is actually bi-directional and implemented in both forward and reverse channels.
- The test software does not simulate silence periods. This means that the tests are actually simulating an extreme case, with the maximum load present at all times on each processor core.

Table 3: Effnet ROHC™ benchmark hardware and software environment

	Intel® Pentium® 4 processor (Based on Intel NetBurst® microarchitecture)	**Dual-Core Intel® Xeon® processor LV 5138^Δ (Based on Intel® Core™ microarchitecture)**
Vendor/Model	Dell Dimension* 8400 Desktop	Intel NetStructure® MPCBL0050 Single Board Computer (AdvancedTCA*)
BIOS Version	Dell Dimension System BIOS (version A09)	Version 1.17.1057: 2006 American Megatrends, Inc. Bios Version: WH500ES0.86E.00.01.0000.092020061136
BIOS Setup	Hyper-Threading Technology (HT Technology) disabled	C1E and Adjacent Cacheline Prefetch disabled
CPU	Intel® Pentium® 4 processor 630^Δ supporting HT Technology† 3.00 GHz	Two Dual-Core Intel® Xeon® processors LV 5138 2.13 GHz
L2 Cache	2 MB	4 MB shared
Chipset	Intel® 925X Express Chipset	Intel® 5000P Chipset/ESB2
Front-Side Bus Speed	800 MHz	1066 MHz
Memory	Dual Independent Memory Channels Samsung 2x 512 MB DDR2 533 MHz CL4 non-ECC* (1 GB total)	Four Independent Memory Channels FBDIMMs Micron 4x 1 GB DDR2 667 CL5 ECC* (4 GB total)
Operating System Kernel	Linux* 2.6.11.4-20a-smp	Red Hat Enterprise Linux AS* release 4 (Nahant Update 4) Linux 2.6.9-42.ELsmp #1 SMP
GNU C Compiler	Version 3.3.5	Version 3.4.4
Intel® C++ Compiler for Linux*	9.1.036 (32-bit)	9.1.036 (32-bit and 64-bit)
ROHC Software	Effnet ROHC™ version 2.2	Effnet ROHC™ version 2.2
ROHC Load Generator Software	Identical versions	Identical versions

Test Environment

The test system environment is shown in Table 3.

Hyper Threading Technology† (HT Technology) was disabled on the Dell system (based on the Intel Pentium 4 processor) because the test application is single threaded and disabling HT Technology allowed it to access the entire L2 cache. Two BIOS settings on the Intel NetStructure MPCBL0050 Single Board Computer were modified. Disabling adjacent cache-line memory pre-fetch on the dual-processor system improved performance of the test application with almost no sequential memory accesses since this type of workload does a high proportion of random-memory accesses. C1E was also turned off. Enhanced C1 state (C1E) is an automatic, very fine-grained voltage/frequency reduction that takes place when an operating system places the processor in a C1 state. C1E was turned off to ensure a consistent processing environment during the testing process. The memory size difference between two platforms should not affect performance, as Effnet ROHC requires less than a Gigabyte of memory even when supporting large number of flows at high processor core loads. It is also not expected that the differences in GNU C Compiler or Intel® C++ Compiler versions made a significant difference in the final results.

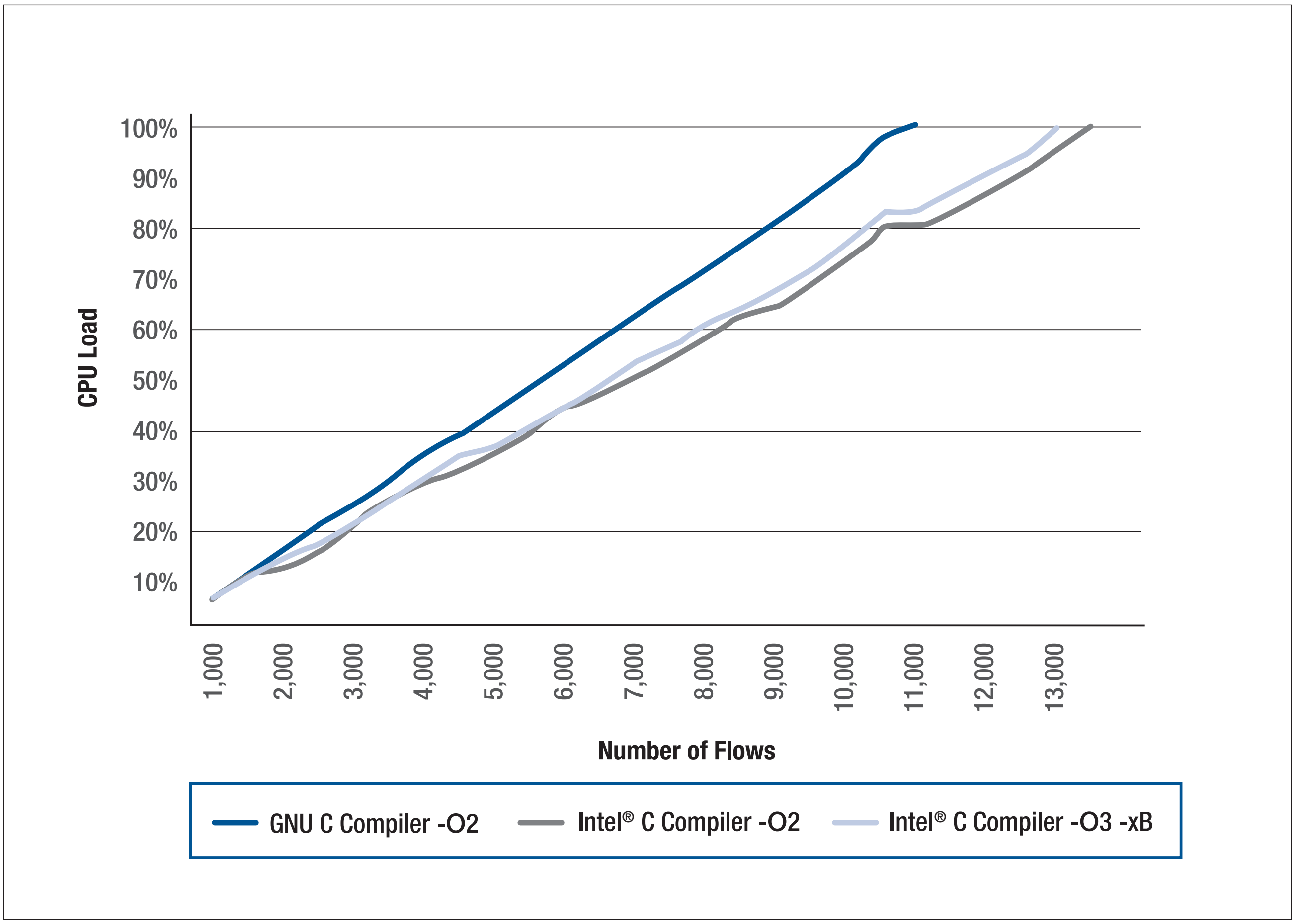

Figure 5: Effnet ROHC™ benchmark results for the Intel® Pentium® 4 processor 630

Effnet ROHC™ Performance on the Dell Desktop Based on the Intel® Pentium® 4 Processor

Testing was carried out with Effnet ROHC version 2.2 compiled with the following C compilers and optimization flags:

- GNU C Compiler -O2
- Intel C++ Compiler -O2
- Intel C++ Compiler -O3 -xB

The results shown in Figure 5, on the previous page, can be summarized as follows:

- The CPU load increases almost linearly as the number of input flows increases. At 50% CPU load, the Intel Pentium 4 processor-based system can handle about 7100 pairs of bi-directional flows.
- The Intel C++ Compiler for Linux with -O2 optimization gives the best performance. The performance improvement varies between 5–20 % over the GNU C Compiler. The higher the number of input flows (the more processor load), the bigger the improvement when the Intel C++ Complier object code was used. In other words, at a lower number of flows the improvement was 5%, but at the highest number of flows the improvement was 20%.
- The Intel C++ Compiler with more aggressive optimization (-O3 -xB) did not give better performance on the Effnet ROHC application.

Effnet ROHC™ Performance on the Intel NetStructure® MCBL00050 Single Board Computer with Dual-Core Intel® Xeon® Processor LV 5138

Two types of tests were conducted on the Intel NetStructure MPCBL0050 Single Board Computer. One compared the GNU C Compiler and the Intel C++ Compiler with optimization flags, and the second analyzed the performance of Effnet ROHC on different numbers of cores.

Compilers and Optimizations

The main purpose of the first group of tests on the Intel® single-board computer was to determine which compiler and compile flag combination gave the best performance. The following compilers and compile option flags were tested:

- GNU C Compiler: m32 -O2
- GNU C Compiler: m64 -O2
- Intel C++ Compiler (32-bit) -O2
- Intel C++ Compiler (64-bit) -O2
- Intel C++ Compiler (32-bit) -O3 -xB

The results are shown in Figure 6 and summarized below:

- The 64-bit Intel C++ Compiler with the -O2 option gave the best performance. It improved performance by approximately 5–15% (in absolute terms) compared to the 64-bit GNU Compiler. It also gave 5–20% better performance compared to the 32-bit Intel C++ Compiler, and 5–30% better performance compared to the 32-bit GNU Compiler. Again, the higher the number of input flows (i.e., the more processor load), the more improvement was noted using the Intel C++ Compiler for Linux.
- At 50% processor load, a single core of the Dual-Core Intel Xeon processor LV 5138 can handle up to 8300 pairs of bi-directional flows when using the 64-bit Intel C++ Compiler for Linux. This is about 18% better than the 7100 pairs that can be handled on the Intel Pentium 4 processor.
- Intel C++ Compiler (32-bit) with normal optimization (-O2) has slightly better performance compared to more aggressive optimization (-O3 -xB).
- The processor utilization increased almost linearly as the input number of flows increased.

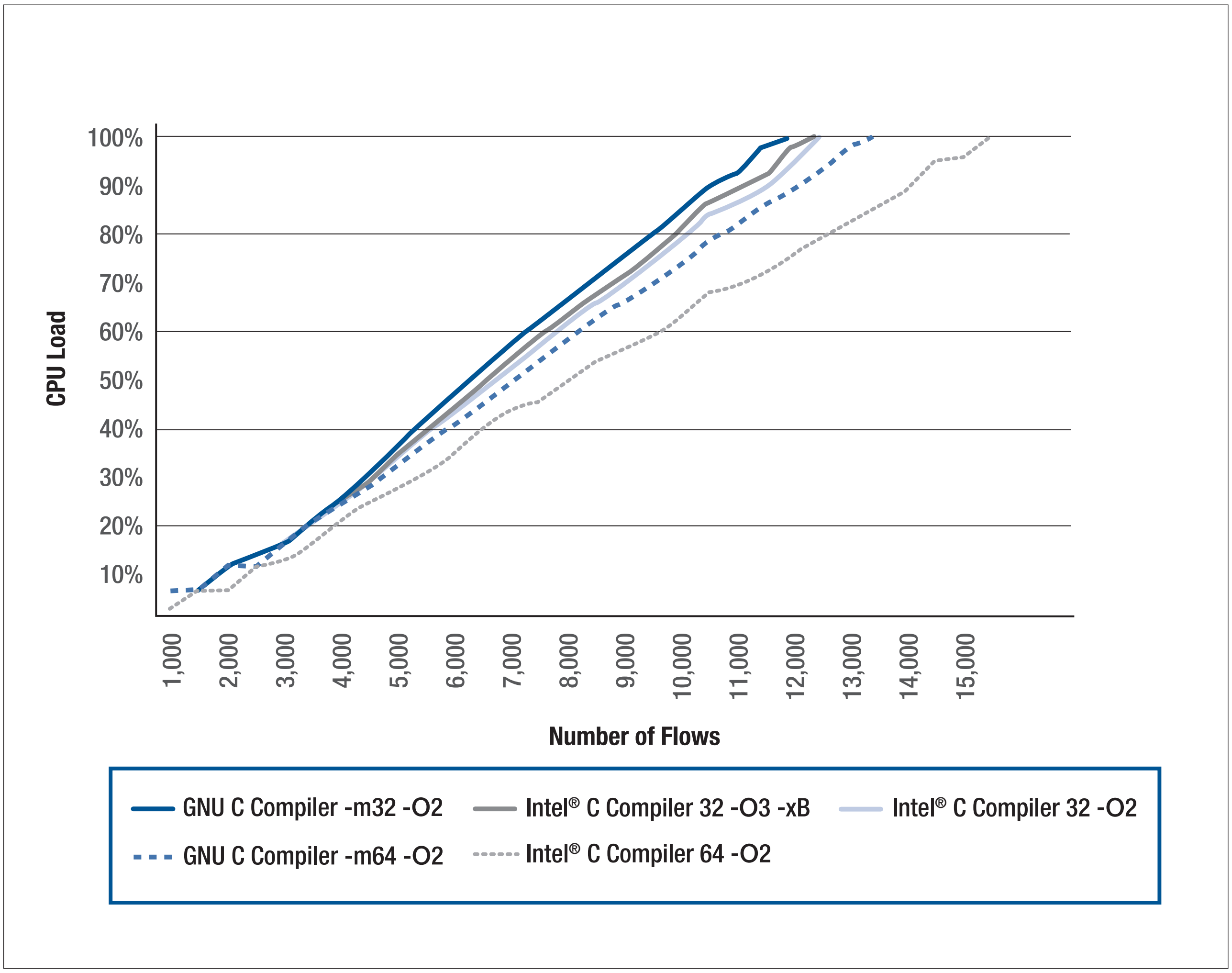

Figure 6: Effnet ROHC™ benchmark on Dual-Core Intel® Xeon® processor LV 5138 (single core)

Scaling Across Multiple Cores

The second series of tests observed the performance difference when one or two processors (1–4 cores) were involved in ROHC processing.

The results are shown in Figure 7 and summarized below:

- As expected, the highest number of flows per core occurred when only a single core per processor was in use (see gray, dotted line in Figure 7).
- The average processor load and performance was the same when the Effnet ROHC application was executing on one core in each processor even though the two processors share a single front-side bus. This suggests the cores are not competing for the front-side bus bandwidth. Subsequent measurements with the Intel® VTune™ Performance Analyzer confirmed this observation.
- The average processor load per core when three processes ran in three cores (see black line in Figure 7) was similar to when two processes ran on two cores in the same chip (see gray, solid line in Figure 7), and also when applications ran on all four cores. This indicated that processes or threads running on different cores had little effect on each other.
- There was an approximate 10% reduction in performance of two cores in the same chip than two cores in different processors. This can be explained due to a slight processing penalty when two cores running separate single-threaded applications share the same L2 cache. In all cases, performance increased linearly as the number of flows increased, regardless of whether 1, 2, 3, or 4 cores were in use. This increase is independent of the number of processor cores.

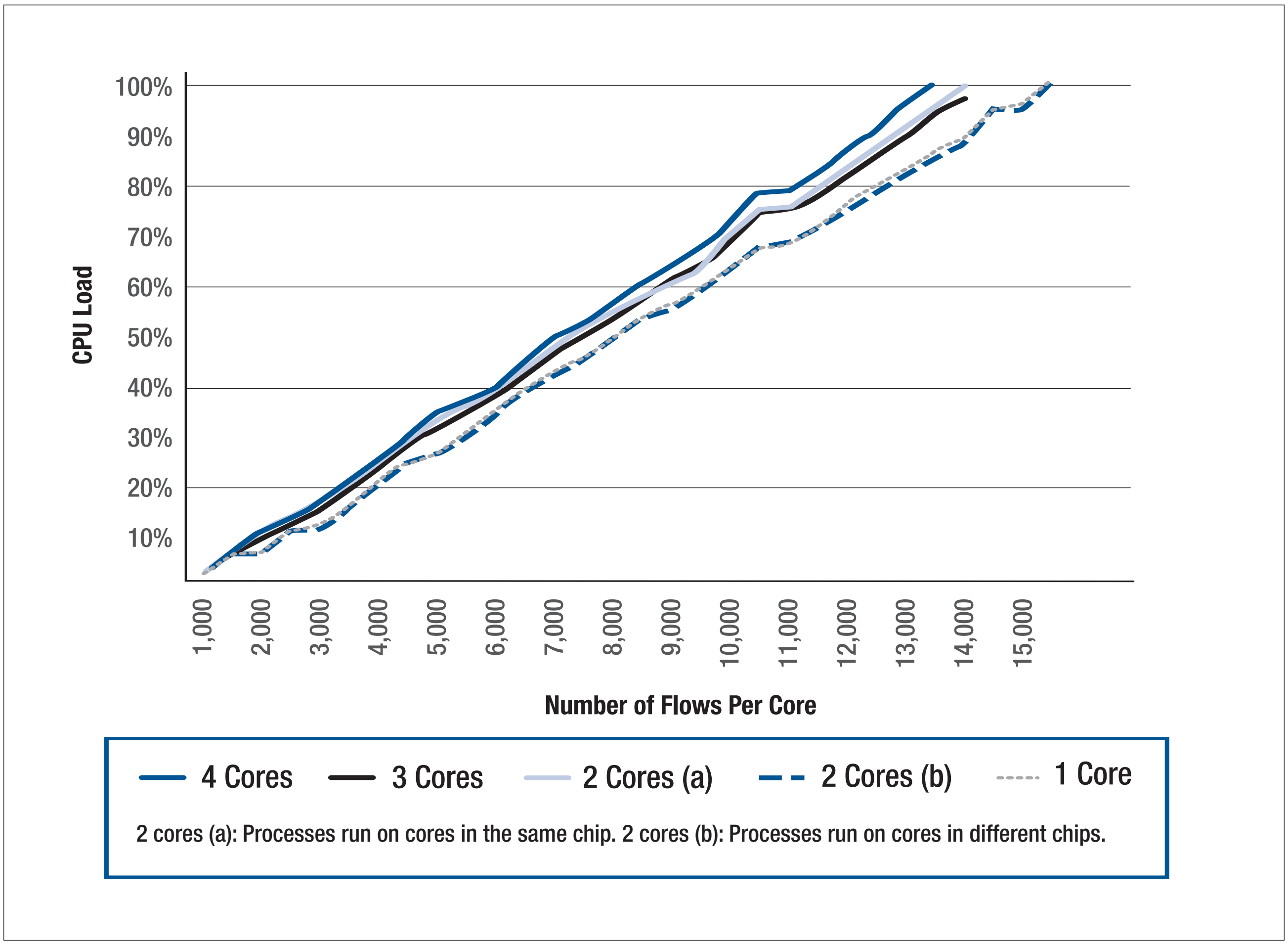

Figure 7: Effnet ROHC™ performance on Intel NetStructure® MPCBL00050 Single Board Computer with dual Dual-Core Intel® Xeon® processors LV 5138 (1-4 cores)

Key Findings

- The Intel C++ Compiler for Linux produced better results than the GNU C Compiler on both platforms.
- The standard optimization flag -O2 resulted in the best performance.
- The Intel VTune Performance Analyzer proved to be a valuable tool to find and remove performance bottlenecks. It enabled performance improvement of about 17% over the latest Effnet ROHC version.
- Utilizing the Intel Pentium 4 processor-based platform, the Effnet ROHC supported about 4500 bi-directional VoIP flows at 30% processor load and about 7100 bi-directional VoIP flows at 50% processor load.
- On the platform using the Dual-Core Intel Xeon processor LV 5138, Effnet ROHC supported about 5700 bi-directional VoIP flows at 30% utilization of one core and about 8300 bi-directional VoIP flows at 50% on a single core.
- Running on multiple processors and cores, Effnet ROHC performance showed the same linear increase pattern as on a single processor. The performance for two cores was slightly better when the cores were in different processors compared to when they were in same processor. Almost linear scaling across multiple cores was observed as multiple cores were used and performance improved. For example, when using two cores at 30% load/core, Effnet ROHC supported about 11400 bi-directional VoIP flows.

Table 4: Benchmark results summary for Effnet ROHC™ on Intel® processors

	Processor Utilization	**Intel® Pentium® 4 processor 630^Δ (Based on Intel NetBurst® micro-architecture)**	**Dual-Core Intel® Xeon® processor LV 5138^Δ (Based on Intel® Core™ microarchitecture)**	**Approximate Scaling Factor compared with 1 core (per core)**
1 Core	30% 50%	4500 7100	5700 8300	
2 Cores (Same Processor)	30% 50%		9600 (4800 per core) 15000 (7500 per core)	84% 90%
2 Cores (Different Processor)	30% 50%		11400 (5700 per core) 16600 (8300 per core)	100% 100%
3 Cores	30% 50%		15000 (5000 per core) 23100 (7700 per core)	88% 93%
4 Cores	30% 50%		19200 (4800 per core) 29600 (7400 per core)	84% 89%

Advantages of Intel® Core™ Microarchitecture for ROHC Workloads

The Effnet ROHC performed 20% better on a single core within the Dual-Core Intel Xeon processor LV 5138 than on a single-core Intel Pentium 4 processor, even though the Intel Pentium 4 processor was running at about a 29% faster clock rate. This performance gain was attributed to the different chipsets in the test systems. The Intel® E5000P chipset on the Intel NetStructure MPCBL0050 Single Board Computer had a 25% faster front-side bus, utilized quad memory channels rather than dual memory channels, and supported 21% faster memory than the Intel® 925X Express Chipset. However, the performance gains appear to be beyond what would have been expected from the more capable memory sub-system alone. It appears that the new Intel Core microarchitecture brings a high degree of branching and frequent, small, random memory accesses to typical user plane workloads characterized by integer and logical operations. The Intel Core microarchitecture pipeline design, for example, may be playing a significant role. It consists of 14 stages and implements both macro- and micro-operations, whereas the Intel NetBurst microarchitecture pipeline has 31 stages. The high degree of branching and the small, random memory accesses of ROHC workloads indicate that a pipeline is less likely to be correctly filled with speculative data and instructions, therefore a time/processor cycle penalty is incurred by having to flush and re-fill the longer pipeline. The Intel Core microarchitecture pipeline is able to execute and return four instructions simultaneously; the Intel NetBurst microarchitecture pipeline can handle only three. The Intel Core microarchitecture also implements advanced branch prediction and advanced memory disambiguation, which feed instructions and data to each core's execution engines more efficiently.

Additional studies would be required to assess whether the larger L2 cache in the Dual-Core Intel Xeon processor LV 5138 contributed to the improved performance. ROHC workloads do not use frequently accessed large data structures like other workloads that typically benefit most from larger L2 caches. Since the Effnet ROHC implementation is single threaded, with each instance bound to one specific core, it could not make use of the highly efficient data exchange between cores enabled by the large, coherent L2 cache implemented in the processor.

Intel® Intelligent Power Capability is a key feature of the Intel Core microarchitecture that played a pivotal role in enabling high-performance scalability in a thermal environment such as those required by AdvancedTCA* systems. The extremely high-processing-per-watt efficiency of the Dual-Core Intel Xeon processors LV 5138 enabled the Intel NetStructure MPCBL0050 Single Board Computer to be built with two dual-core processors, a high-performance Intel® chipset and up to 16 GB of ECC RAM while maintaining the thermal headroom required for placing an Advanced Mezzanine Card on the baseboard.

Acronyms

3GPP: 3rd Generation Partnership Project

3GPP2: 3rd Generation Partnership Project 2

AMR-NB: Adaptive Multi Radio-Narrow Band

ATCA: Advanced Telecom Computing Architecture

BER: Bit Error Rate

CDMA: Code Division Multiple Access

CDMA2000: Code Division Multiple Access 2000

CPU: Central Processing Unit

CRC: Cyclic Redundancy Check

DP: Dual Processor

EV-DO Rev A: Evolution-Data Only Revision A

ESP: Encapsulating Security Protocol

GGSN: Gateway GPRS Support Node

GPRS: General Packet Radio Services

GSM: Global System for Mobile communication

IETF: Internet Engineering Task Force

IMS: IP Multimedia Subsystem

IP: Internet Protocol

L2: Layer 2

PDSN: Packet Data Support Node

QoS: Quality of Service

RFC: Request for Comments

ROHC: Robust Header Compression

ROHC-LLA: Link Layer Assisted-ROHC

RTP: Real-Time Protocol

RTT: Round-Trip Time

TETRA: Terrestrial Trunked Radio

UDP: User Datagram Protocol

UMTS: Universal Mobile Telecommunication System

VoIP: Voice on Internet Protocol

W-CDMA: Wideband Code Division Multiple Access

W-LSB: Window-based Least Significant Bit

Company Information

About Effnet

Effnet AB, a wholly owned subsidiary of the publicly traded Effnet Holding AB (publ), is a world leader in the area of IP Header Compression. Effnet develops and sells embedded software that increases the efficiency, speed and quality of IP traffic in fixed, mobile and satellite networks.

For more information about Effnet and IP Header Compression please visit www.effnet.com.

About Intel

By enabling more content, mobility and capabilities than ever before, Intel gives you the advantage in a rapidly changing world. With advanced silicon building blocks, industry-standard platforms, modular infrastructure solutions and ecosystem support, Intel can help you deliver a more compelling digital lifestyle. Intel, the world leader in silicon innovation, develops technologies, products and initiatives to continually advance how people work and live.

For more information about Intel, visit www.intel.com/pressroom.

EFFNET

intel®

[1] Adapted from the more detailed explanation in Effnet AB white paper, "The Concept of Robust Header Compression, ROHC," February 2004. http://www.effnet.com/sites/effnet/pdf/uk/Whitepaper_Robust_Header_Compression.pdf.

[2] C. Bormann, et al., "Robust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed," RFC 3095, July 2001.

Effnet AB, "The Concept of Robust Header Compression, ROHC", white paper, Feb,2004. http://www.effnet.com/sites/effnet/pdf/uk/Whitepaper_Robust_Header_Compression.pdf

[Δ] Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

[†] Hyper-Threading Technology requires a computer system with an Intel® Pentium® 4 processor supporting HT Technology and a HT Technology enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use. See www.intel.com/homepage/land/hyperthreading_more.htm for additional information.

0107/CHA/OCG/XX/PDF Order Number: 316160-001US