RCTP: A Low-complexity Transport Protocol for Collecting Measurement Data

Péter Orosz, Tamás Skopkó, Máté Varga

***Abstract*— Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are the essential transport protocols of the Internet Protocol (IP) networks. Both of them are dedicated for certain purposes and face their obvious limitations. Thus, there is a constant endeavor for developing alternatives. Despite the reliability feature of TCP, its relatively high complexity does not always enable to implement it in a hardware environment with constrained resources. Our paper introduces a low-complexity transport protocol dedicated to a real-time network monitoring system operating at 10+ Gbps. Its task is to transport the preprocessed IP packets from the monitoring device to the post-processing hosts without loss over a dedicated LAN. Resource requirement on sender side has to be reduced as much as possible while trying to maintain high throughput. Although RCTP is intended to serve in a network measurement system, it may be suitable for other measurement infrastructures such as sensor networks, where data provided by the sensors with limited resources have to be collected at a central node.**

Keywords- Transport protocol; datagram; UDP; monitoring system.

I. INTRODUCTION

Passive traffic monitoring is an essential task for network infrastructure providers. There is a growing demand on high processing capacity to monitor core networks. In order to support lossless packet-level monitoring, dedicated hardware-accelerated equipments should be deployed on certain links of the infrastructure. Besides network processors, an increasing number of solutions are based on Field Programmable Gate Arrays (FPGA) as central building block. This technology ensures flexibility for integrating services into one device and performing real-time processing, while its resources (e.g., number of logical units and available on-chip memory) are always limited. The transport protocol introduced in this paper is optimized to distributed measurement architectures with constrained hardware resources and is part of a high performance monitoring system using FPGA-acceleration for packet capturing, parsing and filtering.

Each probe device is tapping one specific direction of a 100 Gbps link with passive optical splitter as seen in Fig. 1. Received packets are preprocessed within the FPGA. These tasks include high precision timestamping, protocol parsing and classification of the packets.

Manuscript received June 19, 2014, revised August 31, 2014. P. Orosz, T. Skopkó and M. Varga are with the Faculty of Informatics, University of Debrecen, Debrecen, H-4028 Hungary, (emails: oroszp@unideb.hu, skopkot@unideb.hu, varga.mate.unideb@gmail.com).

Packet flows are also identified and forwarded to collector agents, where deeper inspection, heuristic recognition and statistical analysis are done. Up to 10 collectors are connected with 10 Gbps links to one probe device and the monitored traffic of the 100 Gbps link is distributed uniformly between the 10 Gbps links. One specific flow is forwarded to one specific collector only. The transport protocol introduced is intended to serve the transmission of captured traffic from the probe to the collectors.

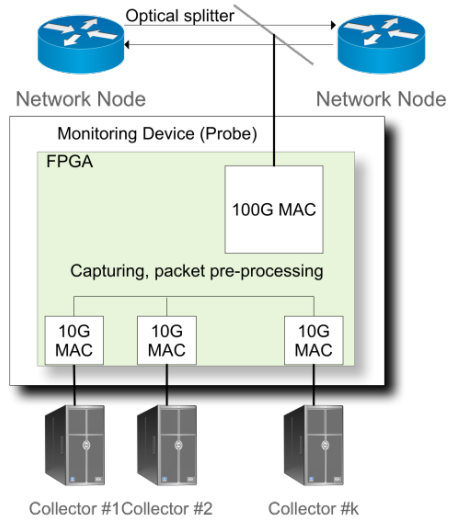

Figure 1. The monitoring system tapping a 100 Gbps link and distributing monitored data to collectors after preprocessing tasks are done

Nevertheless, the proposed protocol can be integrated to any measurement infrastructure where multiple monitoring devices are generating and transmitting measurement data towards processing agents over the network. In an FPGA-accelerated system, the high processing capacity is achievable by using pipelining technique, hence a number of modules (interface MACs, transport protocols, etc) are placed into the chip in multiple instances [1]. Therefore, a transport protocol that is implemented in hardware should have a low resource usage. Many TCP variants provide reliable transmission with high throughput [2][3][4][5][6][7]. However, implementing multiple instances of the protocol inside an FPGA chip results in a significant resource utilization. In our measurement system the captured and preprocessed traffic of an aggregated 10+ Gbps link is distributed to multiple 10 Gbps outgoing connections towards post-processing hosts. Considering the required

memory, TCP retransmission buffer has to be implemented in an off-chip RAM. However, accessing an external memory by many concurrent TCP instances would build up a serious bottleneck at line rate. To overcome processing bottlenecks we commonly used offloading techniques in our former research projects; converting the complex and resource-intensive operations to delay-tolerant processes generally improve the overall performance [8]. In our system the monitoring device itself is connected to the agents directly or through a dedicated LAN. This private infrastructure can forward packets reliably. In our design, transmission control is offloaded from the sender to the receiver in the form of rate control. Therefore, sender side is simple enough to be implemented even in a resource- and energy-constrained hardware. We improved its performance further by packetizing the data into MTU-sized packets before transmission, thus reducing packet and interrupt rates at the receiver.

In Section II, TCP, the most feature-rich and widely applied transport protocol is set in contrast with a more lightweight UDP as well as a number of alternative protocols with reliability feature. Linux packet processing is reviewed from point of lossless packet reception in Section III. In Section IV our custom protocol is introduced in detail. Section V is about the performance evaluation of the protocol using a hybrid FPGA and Linux implementation at 10 Gbps.

II. RELATED WORKS

TCP is a set of algorithms ensuring the adaptation of the protocol to various traffic situations. Reliable transmission is its basic feature, which is implemented using a positive acknowledgement mechanism. Buffering unacknowledged data makes possible to retransmit the whole buffered byte stream or a part of it when packet loss occurs. Researches proposed formulas for optimal sizing of buffers which is commonly specified by the bandwidth-delay product, to increase the performance of the protocol [9]. The size of the retransmission buffer at 10+ Gbps bandwidth, adjusted with this formula, exceeds the megabyte limit. This renders an efficient hardware implementation difficult to design. The complex logic required by TCP's algorithms make the situation even challenging. T. Uchida implemented TCP in FPGA for Gigabit Ethernet [10]. His circuit took up about 3000 slices. It uses block RAM buffering, which is sufficient only for storing small amount of data. We ruled out this implementation since our application requires wire speed transmission at 10+ Gbps, which implies large memory buffers.

By its simplicity, UDP is an optimal basis for a transport protocol dedicated for hardware implementation. Reliability feature is normally implemented in an upper protocol layer or in the application itself, by using extra buffer. UDP Lite is a variant of UDP that enables the forwarding of corrupted packets as well. One of the first attempts of combining simple message-oriented communication with reliable transmission into a lightweight transport protocol was Reliable UDP (rUDP) developed at Bell Labs [11]. It resembles TCP's retransmission mechanism with a windowing technique very similar to TCP resulting in the demand on extra buffer. Although Cisco and Microsoft have their own implementation of this protocol, it remained in draft status at IETF.

Y. Gu and R. L. Grossman worked out an UDP-based transport protocol for WAN purposes [12]. It uses acknowledgements similar for TCP but it is timer-based and makes possible to acknowledge more frequently on slow or unreliable connections. Its main bottleneck for our application is its requirement for sender buffer.

E.Kohler *et al.* introduced Datagram Congestion Control Protocol (DCCP) [13]. It provides a TCP-like feature set with reliable packet transmission and focuses on congestion control using Explicit Notification Control (ECN). Though it does not support in-order delivery of packets, it is not an explicit requirement for our application since packets are timestamped by the monitor hardware. Its real bottleneck is the recovery process triggering by a series of lost packets that vindicates the demand for buffering high amount of data.

An effective combination of TCP and UDP called Stream Control Transmission Protocol (SCTP) was presented by R. Stewart and C. Metz [14]. It supports fragmentation of transmitted data into chunks and thus enabling to overcome some limitations of TCP by making some of its features disabled. It can reliably transmit data but since no system bottlenecks are taken into account, it retransmits lost data. This makes data buffering necessary and thus this protocol is also ruled out.

The same limitation is present if we consider Reliable Datagram Sockets (RDS) developed at Oracle [15]. This transport protocol is designed for inter-process communication (IPC) on InfiniBand and primarily benefits from bidirectional dataflow because acknowledgements are placed at the end of data payload. Although it avoids losses caused by a saturated socket buffer, it does not care about other factors such as processing capacity bottlenecks.

Scalable and Secure Transport Protocol (SSTP) is intended to serve as a scalable transport for metering systems [16]. The protocol presented by Y.J. Kim *et al.* ensures reliable transfer of data for short flows as well as data encryption. Although it is less complex compared to TCP it still contains functions (i.e., fairness and secure layer) unnecessary in our environment.

Y.J. Kim and M. Thottan also presented Smart Grid Transport Protocol (SGTP) as an alternative transport protocol for collecting periodic measured data but it focuses on short-term flows consisting of small packets [17]. Our monitoring system produces long flows and large packets.

Lossless transmission of metered or monitored data through various network conditions (e.g. 802.11x, etc.) is a general requirement. Thus transport protocols intended for these applications generally include buffering and retransmission of lost data. This is not a problem when only short packet flows are present. At the same time it is not likely to achieve line rate transmission performance. Probes collecting the measurement information are generally low performance hardware with a requirement of low resource consumption.

In contrast, data center transport protocols take aim at high throughput besides the demand on reliable transmission. Generic hardware is applied in this environment with the feasibility to use more complex protocols and larger retransmission buffers. Replacement of TCP is researched to achieve better transmission performance.

Our monitoring system differs from the aforementioned monitoring and data center applications. Although information flow is unidirectional, network probes can collect high amount of measurement data. Even though FPGAs have high processing performance, their resources are limited. Amongst monitoring and preprocessing functions, many instances of a chosen transport protocol have to be compiled in. This transport protocol has to occupy as low resources as possible. Throughput has to be maximized and no loss of measured data should occur.

Like in most data centers we also have a dedicated packet forwarding infrastructure that ensures lossless transmission. This implies that a packet loss can be caused by the collector's operating system or data processing application. Also there is only one packet flow per agent, hereby no flow will be interfering and no fairness has to be implemented.

Since we did not find any suitable protocol to fit our special requirement, we created a new datagram-based protocol. The design was driven by the sender-side simplicity and reception-side flow control.

III. BACKGROUND

In a packet processing system packet loss occurs when any of the buffers in the data path gets saturated. There are three critical buffers for UDP reception in the Linux kernel. Since these buffers are critical factors from the view of our protocol, we inspected their behavior more closely. The first one is related to the network interface card (NIC) hardware, it is the RX ring buffer [18]. This buffer is relatively small and its size is fixed or limited to a certain range, if configurable at all. A high performance NIC and its driver can process this queue at wire speed therefore no extra attention needed to take when using enterprise grade NIC such as Intel's server cards. Modern NICs can handle multiple RX queues. Each RX queue can be assigned to a different flow. In our measurement system there is only one packet flow to a specific agent, consequently advances of multiple RX queues cannot be exploited. Moreover current packet processing architectures in the generic operating systems are not designed to parallelize the processing of one packet flow between multiple CPU cores. However utilizing multiple cores is an important factor for preparing our architecture.

The second RX buffer is the ingress buffer or backlog queue. Packets are copied to this queue from the RX ring by the interrupt handler. Modern NAPI based drivers omit this step, hardware interrupts (hard IRQs) are performing critical tasks instead. Our protocol implementation is designed for NAPI supported NICs, thus no consideration is made to the load of the backlog queue. In case of NAPI drivers executing non-critical packet processing tasks are up to the software interrupts (soft IRQs), hereby minimizing the CPU time spent in hardware IRQ context. Moreover soft IRQs can be run simultaneously (with the limitation of single flow per CPU). The most important constraint for this kernel task is the processing capacity of the CPU core executing the soft IRQ for the corresponding RX interface (soft RX). On contemporary generic PC architectures at 1 Gbps or higher data rate, processing high intensity traffic containing small sized packets saturates the CPU. To maximize the performance of these architectures, it is advisable to use MTU-sized packets, especially if latency is not a critical factor.

The last kernel-related buffer is the sockets' receive buffer. A user space application receives data through this queue. The socket buffer is not a ring buffer. If the application cannot keep up with processing, the kernel will drop packets not fitting into the socket buffer. Size and occupancy of this buffer have to be considered if reliable transmission is an aim.

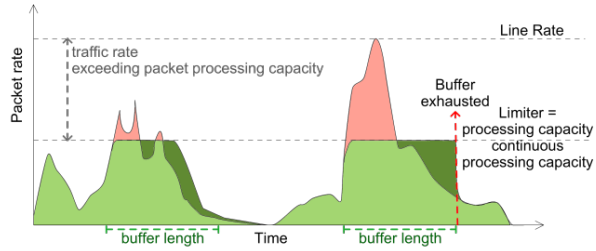

Figure 2. Packet processing performance behaving like a traffic shaper

Performance of software based packet processing is strictly limited by the processing capacity of the CPU core running the processing task. Intensive bursts overwhelm the processing CPU core. In this case, a smaller amount of packets are processed immediately, while the remaining packets can be processed when no new packets arrive. By using larger buffers the number of these temporarily unprocessed packets can be increased. When processing longer buffers we can also expect higher latency in packet forwarding. Also, larger buffers compensate processing bottleneck temporarily. In this aspect, packet processing subsystem of the operating system acts like a traffic shaper as seen in Fig. 2 where CPU's processing capacity corresponds to a limited bandwidth. If there is a higher rate of traffic at its input, it lengthens the traffic in time with a lower de-queuing rate.

We also examined the Ethernet flow control mechanism as a possible method for controlling the packet rate. We concluded that it is not efficient enough because it allows only a pulse-width modulation (PWM) type shaping of the traffic and control packets should be sent out at a very high rate.

IV. INTRODUCTION TO RCTP

The control mechanism in RCTP is based on a closed-loop controller. Fig. 3 shows the feedback mechanism of the protocol. Sender is instructed by the receiver to send packets at a specified rate. The rate is adjusted by a control function based on resource usage. Sender side (inside a monitoring

hardware) is low-complexity. Its task is the send UDP packets at the current limiting rate. Software-based reception side (processing agent) is not strictly limited in resources, thus it can implement and handle more complex tasks. We have chosen the generic Linux operating system as implementation ground.

Continuous monitoring of critical system resources enables us to avoid packet loss. As we unfolded in Section III, the most important metrics are the current usage of the RX socket buffer and the current load of CPU core running the soft RX. These parameters are monitored at a fixed sampling frequency specified at the initialization phase. These values are fed into the controller function. For output, the function calculates an enforced packet rate that the agent is able to keep up with the processing considering the current system load and buffer occupancy. Our experiments show that different control functions and parameters are not equally efficient for every metrics. They are detailed in subsection IV.D.

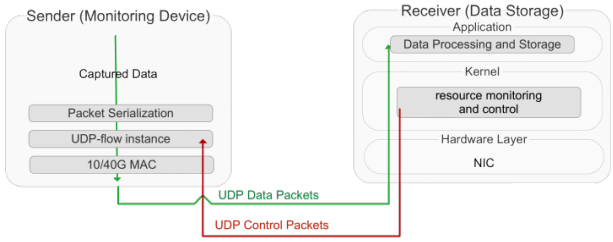

Figure 3. Rate and control mechanism of RCTP transported data

A. Sender-side operation

Data sender implements a simple logic packetizing the byte stream to MTU-sized UDP packets. These are transmitted to the agent. Transmission of a packet is scheduled by the enforced sending rate. Fig. 4 displays transmission loop based on the forced packet rate and the action to take upon reception of a control packet. When a rate control packet is received, a new packet rate is calculated to schedule later packet transmissions. Only a small-size FIFO is required for packing the payload and shaping the traffic to the enforced rate. This can be implemented in block RAMs inside the FPGA.

A timeout parameter built into the packetizer can trigger flushing data from the FIFO even if there is not enough data to construct an MTU-sized packet. The latency caused by packetizing and delayed flushing is not a bottleneck in the monitoring system.

B. Receiver-side operation

Reception side consists of two parts. The receiving application is a user space program capable of receiving UDP packets. Its socket buffer size has to be adjusted considering connection bandwidth and network delay.

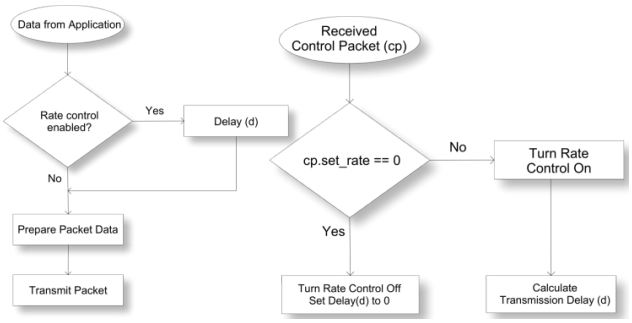

Figure 4. Sender-side packet transmission based on current delay (left) and control packet feedback mechanism (right)

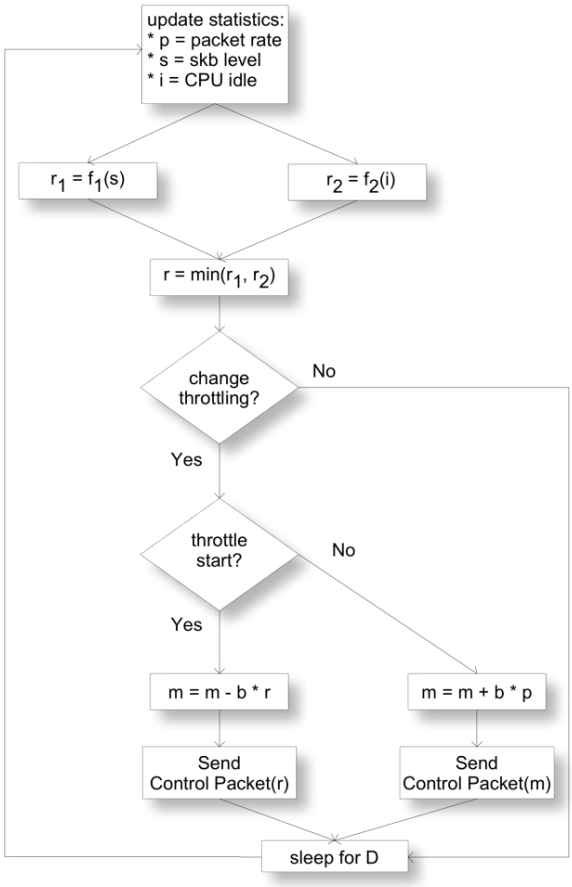

Figure 5. Rate control mechanism operating at the receiver side

Resource monitoring and packet loss avoidance is done by the controller logic. Since its task is time-critical, the controller should be implemented as a kernel process. This ensures a faster feedback. Although we implemented the protocol's control logic in a Linux kernel module, it can be ported to any operating systems capable of reporting the corresponding performance metrics for the controller's input and its exclusive execution can be assured.

Fig. 5 depicts the rate calculation and the signaling mechanism. Current packet rate, socket buffer occupancy

and idle CPU cycles are monitored. Both for the socket buffer usage and the CPU load, the desired packet rates are calculated by the control function (denoted by f_1 and f_2 respectively). The minimum of these rate values is taken as the effective packet rate. If throttling state is changed, control packet will be transmitted. In the throttling start phase, the previously calculated new packet rate is sent out, while in throttling stop phase, sender is informed about the maximal packet rate. The initial value of the maximal packet rate is started at the nominal packet rate derived from the bandwidth. On throttle start, it is decreased based by the product of the new rate and the constant b_1. On throttle stop, it is increased by the product of the current packet rate and the constant b_2. This avoids flapping effect and helps to stabilize throughput. The values for b_1 *and* b_2 were determined empirically (see Section V).

C. Preparing the Operating System

Performance of RCTP depends on certain hardware and operating system factors. Bandwidth is an important factor. As we unfolded in Section III, buffer sizes have to be adjusted considering the link bandwidth and the processing capacity.

The minimum network buffer (skbuff) and socket buffer size can be determined by (1):

$$S = 2 \times B \times \frac{1}{F} \times D / 8 \qquad (1)$$

where S is the size in bytes, B is the connection bandwidth in bps, F is the sampling frequency and D denotes the roundtrip time (seconds), since control information should also reach the sender. Multiplication by 2 ensures the elasticity to compensate jitter caused by scheduling the sampling task.

On a Linux-based reception side, the packet processing related system control (sysctl) parameters to adjust are: *net.core.rmem_default*, *net.core.rmem_max*, *net.ipv4.udp_rmem* and *net.ipv4.udp_rmem_min*.

Optimizing of *net.core.netdev_max_backlog* is not necessary when using NIC drivers with NAPI support. Recent 10+ GbE drivers are prepared to polling mode since high bandwidth cannot be processed efficiently using the traditional 1 interrupt per *k* packet technique.

Modern operating systems utilize an IRQ balancer to distribute soft IRQs among CPU cores efficiently. Its aim is a uniform distribution of soft IRQ load in a multiprocessor system. There are some critical tasks for RCTP that are intended to be explicitly executed on dedicated CPU cores and increases the probability of leaving the information for execution in a lower level CPU cache. This ensures a more predictable scheduling and processing of the corresponding tasks. On a Linux system, the affinity mask specifies the cores that are allowed to run a certain soft IRQ, and can be configured using the *smp_affinity* interface in the */proc/irq* interface tree. For a user space application the *taskset* utility can be used for the same purpose. For an explicit assignment, the *cgroup/cpuset* toolset should be invoked.

The optimal distribution strategy for a multicore system is the following: one core for the RCTP kernel process, one core for the interface's soft RX, one other core for the transmission software IRQ (soft TX), and at least one core is dedicated for the user space application. If soft TX is separated, there is more chance to transmit the control packet with shorter delay resulting in a shorter feedback time.

There is no point for using real-time and preemptive kernel. Since time critical performance monitoring is done in kernel space on a carefully configured system, no user space application will disturb the execution of the control process. Using large buffers eliminates the advantages of a real-time kernel.

Performance metrics feeding into the control function are monitored at a fixed sampling rate. Resolution of statistics is directly affected by the kernel ticks (jiffies), especially for the CPU related metrics. For optimal result a 1000 Hz kernel should be used. This allows a sampling period down to 1 ms. However, an optimal trade-off between sampling frequency and feedback accuracy should be determined. The higher the sampling frequency, the lower the resolution of CPU idle statistics report, resulting lower accuracy of optimal transmit rate prediction. Though reducing scheduling-clock ticks (NO_HZ kernel option) helps to save energy, it deforms CPU idle statistics therefore it should be turned off.

Modern network interface cards support offloading of certain packet processing operations to leave more CPU time for other tasks. These features affect the performance of RCTP also. RX checksumming offloads the verification of UDP checksum to hardware. Enabling this feature is recommended. Since no fragmentation feature is used at the construction of packets, UDP fragmentation offload (UFO) feature cannot be exploited, it can be left disabled. Generic receive offload (UFO) and large receive offload (LRO) allow combining smaller sized UDP packets into a larger, near MTU-sized one. These features cannot be exploited since we are already using MTU-sized packets. Therefore, they would deform the packet counter in the kernel rendering the protocol's control function unusable.

D. Control functions

Current packet rate and monitored performance metrics are input parameters of the control function to calculate the new advertised packet rate.

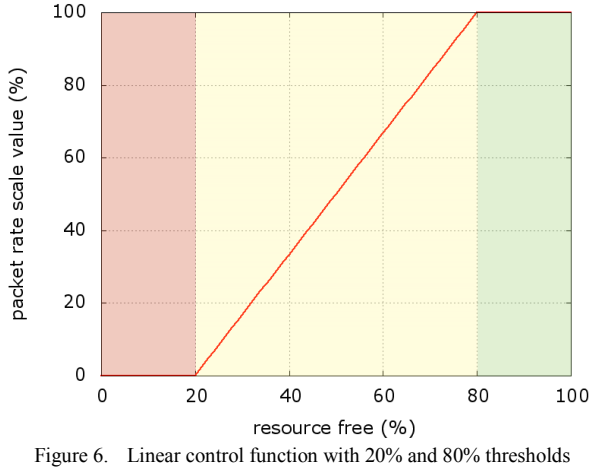

Figure 6. Linear control function with 20% and 80% thresholds

Seven different control functions were evaluated. Each of them has its green, yellow and red intervals as seen in Fig. 6.

In the green range there is no throttling necessary, the sender can operate at an arbitrary packet rate (100% scale value, denoted by *m* in subsection B). In the red range traffic should be immediately stopped (0% scale value). In the middle yellow interval, a new throttling rate is enforced. Fig. 7 shows the shape of the implemented control functions.

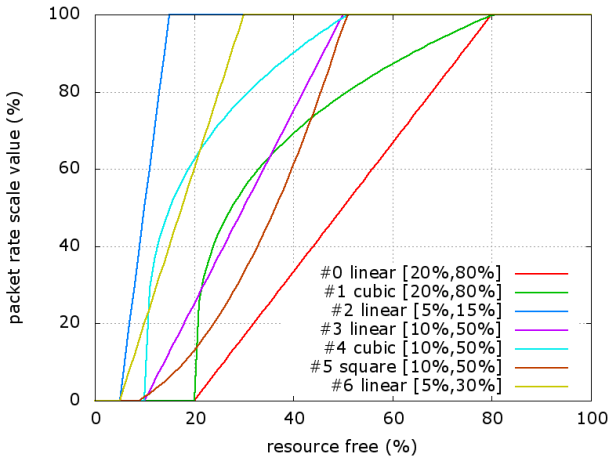

Figure 7. Throttling curves of different control functions

While cubic-based functions throttle less aggressively near the red interval, square-based one throttles more.

The effectiveness of the applied control functions depends on the resolution of the metric the function is applied on. Since socket buffer size is at least in the 10^5 bytes order, metric for socket buffer usage provides higher resolution. We should consider that there is no floating point arithmetic available in the Linux kernel by default, for performance reason we should operate on integer values. Since the most obvious method for monitoring CPU load is to query the number of idle cycles, the resolution (R) of this metric is defined by (2)

$$R = f \,/\, (1000 \,/\, F) \tag{2}$$

where F is the kernel frequency (jiffies) and f denotes the sampling period (in ms).

V. PERFORMANCE EVALUATION

We evaluated the performance of the protocol in a 10 Gbps measurement setup in a configuration similar to the network monitoring system. Sender side was implemented in C programming language (as a user space application) as well as in VHSIC Hardware Description Language (VHDL) on NetFPGA-1G (for testing Gigabit Ethernet performance) prototyping on NetFPGA-10G. Reception side is a generic x64 Linux system running on an Intel Core i7 architecture equipped with an Intel 82599ES 10 GbE NIC. The system was configured as a non-preemptive 1000 Hz server host and critical parameters were set and resource allocation was done as described in Section IV. The controller was implemented as a Linux kernel module and the user space application was implemented in C.

Using the user space application the system was not able to generate a single flow of packets at wire speed even on a 3 GHz CPU. To eliminate this bottleneck, we continued the validation using an FPGA implementation of the sender side.

Our implementation includes a reporting interface for monitoring all the metrics that are inputs of the controller as well as other metrics such as throughput and the rate of soft RX execution.

A. Control functions

Performance metrics are fed into the controller as input parameters. For each metric a desired packet rate is calculated using a pre-defined control function. We evaluated linear as well as nonlinear functions to investigate the effectiveness of the protocol. In the following measurements CPU metric was the most frequent throttling factor. Fig. 8 represents the *'cubic 20-80%'* control function described by (3). In contrast, Fig. 9 shows the behavior of the most effective '*linear 5-30%'* control function described by (4) and activated in the range of t_0=5 and t_{max}=30.

$$F(x) = 25.5 \times \sqrt[3]{x - t_0} \tag{3}$$

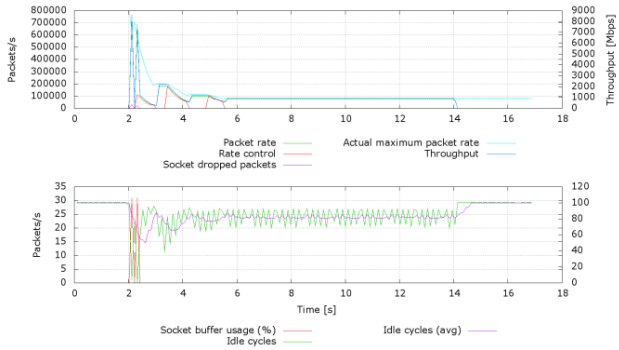

Figure 8. Behaviour of control mechanism using the *'cubic 20-80%'* function

Transmission starts with a close to the wire speed packet rate but after CPU resource drops below the throttling threshold the controller enforces a lower packet rate. After being able to get some more idle cycles it carefully increases the allowed maximum packet rate. It soon reaches the steady state when no rate change is necessary. The achieved throughput is much higher than in the case of the other involved control functions we evaluated, and is comparable to TCP's throughput as well.

$$F(x) = 100 - (t_{max} - t_0) \times (x - t_0) \tag{4}$$

This function is to be evaluated between the transfer stop threshold denoted by t_0 and unconstrained transfer threshold marked with t_{max} formerly.

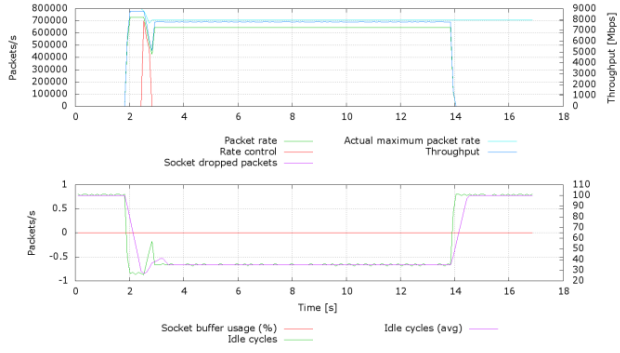

Figure 9. Behaviour of control mechanism using the *'linear 5-30%'* function

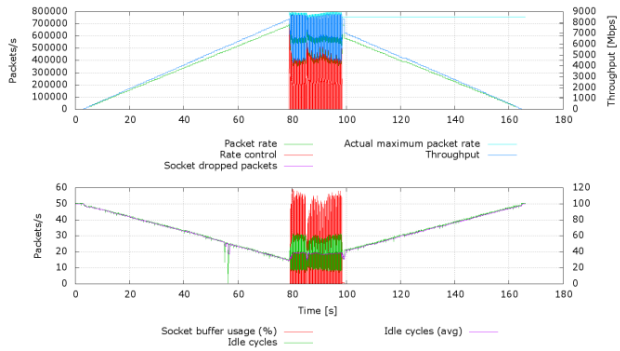

Figure 10. Flapping effect in controller caused by constants b_1=0.1 and b_2=0.1

Fig. 10 shows the flapping effect. During the measurement, MTU-sized packets were transmitted. Packet rate was continuously increased from 1 to $8*10^6$. A '*cubic 10-50%'* control function was used. Using both 0.1 for penalty and premium constant caused a flapping of control and destabilized the packet rate. Using a penalty constant value of 0.2 stabilized the transmission rate (Fig. 11).

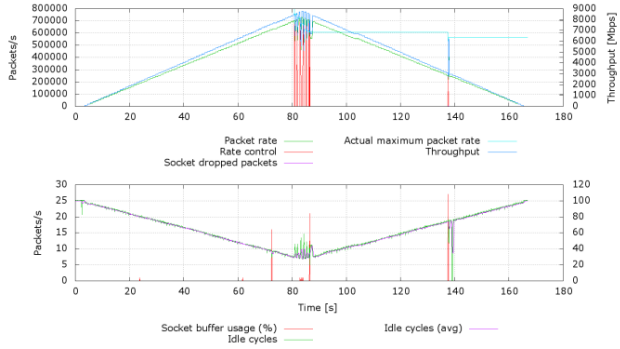

Figure 11. Stabilized throughput using constants b_1=0.2 and b_2=0.1

B. Sampling frequency

As we referred it in Section IV B, the resolution of CPU load metrics is affected by the sampling frequency. The lower sampling frequency the higher metric resolution due to the maximum number of idle CPU cycles between two sampling moments. At the same time, lower sampling frequency implies slower feedback in the control loop. This is analogous with using a longer connection. A lower frequency also implies the need for larger socket buffers. Advantage of higher sampling frequency is the shorter time to converge the maximum packet rate limit. We performed measurements using 10, 25, 50, 75 and 100 ms sampling periods.

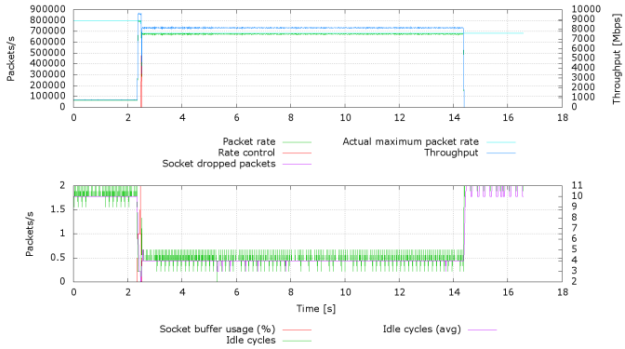

Figure 12. Faster convergence using a higher sampling frequency

Fig. 9 shows a measurement at a requested packet rate of $8*10^6$ and sampling period of 100 ms. The control function was the most effective with the '*linear 5-30%'* configuration. After starting the transmission, idle CPU time immediately dropped. The controller enforced a relatively low packet rate. This resulted in an increased number of idle CPU cycles. This allowed increasing the maximum packet limit. It took about 1 second to reach the steady state at a not significantly lower packet rate than the original one.

Fig. 12 depicts the results of a measurement with the same parameters but at a higher sampling frequency, corresponding to a 10 ms sampling period. This also resulted in as a high throughput as with 100 ms but the convergence time was much shorter.

Although the resolution of CPU metric for the 10 ms sampling period is not as fine as for 100 ms, the protocol's algorithm could efficiently control the packet rate. However, convergence period is shorter while it enables higher throughout.

C. NIC offloading functions

We did not implement UDP checksum generation, thus the performance improvement of RX offloading could not be tested. But in a production system this could be exploited, especially when measurement data has to be collected at higher packet rate.

D. RCTP versus TCP

Although TCP provides a reliable packet transmission at a respectably high throughput, our implementation of RCTP has also a comparable performance. Table I summarizes the most important features of TCP. Despite RCTP does not have such a wide range of abilities, packet losses of the

receivers can be totally avoided in our system. TCP's feature set could be replaced by a sophisticated rate control mechanism.

TABLE I. FEATURE COMPARISON OF TCP AND RCTP

Resource	TCP	RCTP
Session initialization	Yes	No
Reliable transmission	Retransmission, congestion control, flow control	Rate control
Protection against packet loss	Yes	Yes *
Protection against reordering or duplication	Yes	No
Protection against packet corruption	Yes	No
Fairness	Yes	No **
Full duplex	Yes	No

* Packet losses caused by the forwarding infrastructure is not handled
** Not needed because of one flow per agent

In Table II, an enlightened implementation of TCP served in a very similar monitoring system is set in contrast with our RCTP implementation on the same FPGA chip. Even with this simplified design of TCP, the improvement of demand on resources RCTP is significant. Note that more instances of the protocol are present in the production system at the same time, thus multiple savings on resources can be expected.

TABLE II. SIMULATION COMPARISON OF RESOURCE OCCUPATION OF TCP AND RCTP ON THE SAME XILINX XC5VLX110T FPGA CHIP

Resource	TCP	RCTP	Total Available
Number of Slice Registers	1350	449	69120
Number of Slice LUTs	2322	647	69120
Number of Block RAM/FIFO	9	0	148

VI. CONCLUSION AND FUTURE WORK

We introduced a low-complexity transport protocol called RCTP optimized for transporting monitored data to multiple post-processing agents in a monitoring system using low hardware resources considering the resource-constrained design of the data sender. It can avoid packet loss introduced by the receiver, while its throughput can be close to the performance of TCP. Its demand for resources, comparing to TCP, is much lower, which is an important factor for implementing multiple functions in a reprogrammable architecture array such as an FPGA. We evaluated the protocol at 10 Gbps using its FPGA and Linux implementations.

Although it does not seem to be a problematic point, the relatively low resolution of the CPU metric could be improved by an adaptive sampling technique where the sampling frequency could be changed in runtime. By using an initialization phase or by collecting statistics data, an optimal sampling period could be determined. The aim of optimization is to achieve a shorter convergence time or to decrease the time while metrics show the resources being low.

ACKNOWLEDGMENT

The research of Péter Orosz was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP-4.2.4.A/ 2-11/1-2012-0001 'National Excellence Program'.

REFERENCES

[1] W. Jiang, and V. K. Prasanna. "Large-scale wire-speed packet classification on FPGAs," Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays, ACM, 2009.

[2] S. Floyd, "HighSpeed TCP for Large Congestion Windows," RFC Editor, 2003.

[3] L. Xu, K. Harfoush, and I. Rhee. "Binary increase congestion control (BIC) for fast long-distance networks," *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE, 2004.

[4] S. Ha, S, I. Rhee, and L. Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." *ACM SIGOPS Operating Systems Review* 42.5 (2008): 64-74.

[5] K. T. J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," *Proceedings of PFLDnet 2006* (2006).

[6] H. Shimonishi, T. Hama, and T. Murase, "TCP-adaptive reno for improving efficiency-friendliness tradeoffs of TCP congestion control algorithm," *Proc. PFLDnet*. 2006.

[7] A. Baiocchi, A. P. Castellani, and F. Vacirca. "YeAH-TCP: yet another highspeed TCP," *Proc. PFLDnet.* Vol. 7. 2007.

[8] P. Orosz and T. Skopko, "Multi-threaded Packet Timestamping for End-to-End QoS Evaluation," 8th International Conference on Systems and Networks Communications, ICSNC 2013, October 27-31, 2013, Venice, Italy, ISBN 978-1-61208-305-6, Proceeding p. 94-99.

[9] P. Orosz, J. Sztrik, and C. Kim, "Dynamics and Congestion Control of Alternative TCP Variants on Asymmetric Lines," *Communications and Networking*.

[10] T. Uchida, "Hardware-based TCP processor for gigabit ethernet," *Nuclear Science, IEEE Transactions on* 55.3 (2008): 1631-1637.

[11] T. Bova, and T. Krivoruchk, "Reliable UDP protocol, Internet Draft, Network Working Group, draft-ietf-sigtran-reliable-udp-00.txt," February, 1999.

[12] Y. Gu, and R. L. Grossman. "UDT: UDP-based data transfer for high-speed wide area networks," *Computer Networks* 51.7 (2007): 1777-1799.

[13] E. Kohler, M. Handley, and S. Floyd. "Designing DCCP: Congestion control without reliability," ACM SIGCOMM Computer Communication Review. Vol. 36. No. 4. ACM, 2006.

[14] R. Stewart, and C. Metz. "SCTP: new transport protocol for TCP/IP," Internet Computing, IEEE 5.6 (2001): 64-69.

[15] R. Pandit, "Reliable Datagram Sockets (RDS)," OpenIB Developers Workshop, Feb 2006. https://www.openfabrics.org/index.php/ofa-documents/presentations/doc_download/169-ranjit-pandit-silverstorm-reliable-datagram-sockets-rds.html, [retrieved: June, 2014]

[16] Y.J. Kim, V. Kolesnikov, H. Kim, and M. Thottan, "SSTP: a scalable and secure transport protocol for smart grid data collection," Smart Grid Communications (SmartGridComm), IEEE International Conference, 2011.

[17] Y.J. Kim, and M. Thottan, 'SGTP: Smart grid transport protocol for secure reliable delivery of periodic real time data," Bell Labs Technical Journal 16.3 (2011): 83-99.

[18] C. Benvenuti, "Understanding Linux network internals," O'Reilly Media, Inc., 2006.

Péter Orosz received his MSc. in Computer Science from the University of Debrecen, Hungary in 2003 and a PhD degree on traffic analysis of high performance networks in 2010. As research associate he worked in the Network Research Laboratory at Queen Mary, University of London where he was engaged in the performance analysis of network monitoring architectures. Since 2009, he has been working at the Faculty of Informatics, University of Debrecen and currently holds an assistant professor position. In 2010, involving PhD students he established a research group called ATMA focusing on IP network management including high performance network monitoring and traffic analysis using reconfigurable architectures. Currently, he is the leader of the ATMA research group and is participating in several research projects.

Tamás Skopkó received his MSc. in Computer Science from the University of Debrecen, Hungary in 2002. He started his PhD studies in 2009 and from 2010 is a member of the ATMA research group from its foundation. His research focus is at network monitoring and media quality evaluation. He is currently a PhD candidate and was participating in several research projects.

Máté Varga received his BSc. in system engineering from the University of Debrecen, Hungary, where he is currently a master student. Since 2012 he has been working with the ATMA group and participating in a research project focusing on reconfigurable architectures and high performance network measurement.

Internet of Things: Application Areas and Research Results of the FIRST Project

Zoltán Gál, Béla Almási, Tamás Dabóczi, Rolland Vida, Stefan Oniga, Sándor Baran, István Farkas

***Abstract* - The FIRST/IoT project coordinated by the Faculty of Informatics, University of Debrecen, Hungary has important impact on the R&D work in this field. Six activity areas have been covered in the twenty-seven months long project. More than thirty researchers from half dozen Hungarian and other universities and research institutes have been involved in this activity. The results of this work are planned to be used for other international IoT projects in the following time period. Other institutes and individual researchers from abroad are invited to join to this open initiative and become partner. In the paper are presented the results and the most exciting aspects of the research activity.**

***Index Terms* - IoT, MPT, Sensor/Actuator, Big Data, Data Clusterization, Cyber-Physical Space, Bloom-filter, E-health, Ensemble Forecasting, Virtual Organization.**

1. Introduction

Several universities and academic research institutes in Hungary working together with over forty professors and researchers from the United Kingdom, Russia and Romania are involved in effective research activity in the topics of IoT. Faculty of Informatics of the University of Debrecen in Hungary has leader role in the IoT research based on consortium project financed by the EU structural fund and the government of Hungary.

The R&D activity includes six topics: i) Integration of the IoT into the IPv4/IPv6 systems (development and analysis of multipath protocol stack networks; evaluation of L1/L2 transmission mechanisms of the sensor networks; energy usage efficiency of WSNs; analysis of the random fields defined on space-time domain to model the transmission events of radio channels - kriging; cluster analysis of sensor variables; surprise event detection at CEP - Complex Event Processing and ESP - Event Stream Processing supported services; bilateral teleoperation over wireless networks). ii) Cyber physical systems (embedded digital systems and integration of the network technologies; analysis of the complex, real time, dynamic reconfigurable systems; network security and intrusion detection in sensor network critical infrastructures). iii) Self-optimizing and self-managing communication mechanisms of the IoT systems (context dependent addressing for IPv4/IPv6 and 6LoWPAN systems; context dependent clustering, routing and multicast on the IoT; opportunistic networking; context-aware communication for the IoT). iv) E-health powered by IoT (development of intelligent home and vital technologies; real time human activity monitoring; remote supervision; elder people activity recognition; life quality enhancing services; indoor localization techniques using wireless sensor network). v) Weather prediction network tool development and analysis (statistical calibration by BMA and EMOS methods of the temperature and wind velocity ensemble prognosis; analysis of the cosmic background relay with the spectrum of random fields defined on the sphere). vi) Development of testbeds and virtual service platforms (authentication method with two factors and increased security level). In the following chapters the subjects listed above are presented.

2. Integration of the IoT with the IPv4/IPv6 systems

In this topic two R&D fields were included. The importance of the multipath transmission (MPT) of the packet switched technology on network and transport layers was analysed. The effect of the MPT to the IPv4/IPv6 protocol stack was demonstrated by an own developed software library. The other group of tasks was oriented to the statistical analysis of the multicast traffic, to the cluster analysis of the data coming from network with high number of variables and to the frequency resource usage of a supercomputer system.

2.1 The MPT software library

The integration of the IoT with the IPv4/v6 systems opens questions on the efficient bandwidth usage of the available multiple interfaces (e.g. RJ-45, WiFi, 3G, Bluetooth) of the hosts (especially of mobile hosts) especially in the transition process from IPv4 to IPv6. The traditional IP communication infrastructure is restricted to a single IP address (and single interface) usage on the communication endpoints. The IP address is used not only to identify the interface of

[1] Manuscript received June 16, 2014, revised September 8, 2014.

Zoltán Gál, Béla Almási, Stefan Oniga and Sándor Baran are with the Faculty of Informatics, University of Debrecen, Hungary.

Tamás Dabóczi and Rolland Vida are with Inter-University Cooperative Research Centre, Hungary.

István Farkas is with National Information Infrastructure Development Institute, Hungary.

the node, but it is also used to identify the communication session (i.e. socket id). Distributing a communication session between different paths is an interesting question, and it is a focused research area today. Easy to see, that the usage of multiple interfaces and paths will increase the throughput of the communication (see e.g. [1]). If the communication session is terminated on a moving node (e.g. computer located on a moving car) the request of changing the IP address inside a communication session may appear. The traditional L3 roaming solution suffers from the efficiency problem of "triangular inequality". Opening the possibility of changing the IP address of the end node (with the assumption, that the communication session must continue the work), could open a quite new solution area for these situations: The moving computer could easily change its IP address without losing the communication session's state, and this solution could eliminate the triangular inequality problem.

At the Faculty of Informatics, University of Debrecen a software library was created (named as "MPT software library), which opens the possibility of using multiple interfaces (and multiple paths) inside a communication session between the endpoints. The individual paths can be turned off and on without losing the connection. The MPT introduces a new conceptual working mechanism, which differentiates the identification of the communication session (i.e. the socket id) and the identification of the physical interfaces. The solution is based on creating a logical (tunnel) interface on the endpoint. The logical interface is used to identify the node's communication sessions, and it is independent of the physical interfaces. The MPT software library maps the logical interface to multiple physical interfaces dynamically, so offering a L3 multipath working environment. Measurement results show, that the MPT library is able to aggregate the throughput of independent paths very efficiently (see [1], [2]). As the logical interface and the physical interfaces are handled independently, it is also possible to use different IP versions on the logical interface (i.e. by the communication software) and in the physical network environment (see [2]), so the MPT library also offers a seamless IP version changing solution. The detailed description on the MPT library can be found in [3].

2.2. Analysis of the IPv4/IPv6 data traffic and control signals transmitted through the sensor networks

The service effect of the new virtual interfaces based on the new IEEE 1905.1 technology was analysed in PAN/SOHO environment. The current smart devices (tablets, phones, etc.) have multiple physical interfaces with different communication technologies (i.e. Bluetooth, NFC, WiFi, USB) able to communicate concurrently. In the classical protocol stack architecture each interface should have own logical address to communicate simultaneously. A given logical address is mapped to the unique physical address of the interface and each logical address should be placed in separated logical network. Introducing a virtual interface function between the LLC and MAC sublayers, the smart device becomes a switch in the OSI layer L1.75 with only one logical address in the network layer. All the physical interfaces remain active with the own communication technology and participate in the merged group of layer L1.5 channels.

Nice results were obtained by the analysis of the congestion effect to the streaming transmission in low bandwidth, sensor based network environment. It was found that both, the channel load and the channel intensity need to be considered for proper evaluation of the congestion in homogeneous TCP or heterogeneous aggregated TCP/UDP multimedia traffic. The aggregated traffic of the congested streams has long range memory (LRD) characteristic [4].

The coexistence of different wireless transmission technologies (i.e. IEEE 802.11 and IEEE 802.15.4) on the same physical environment was studied in function of the frame size transmitted [5].

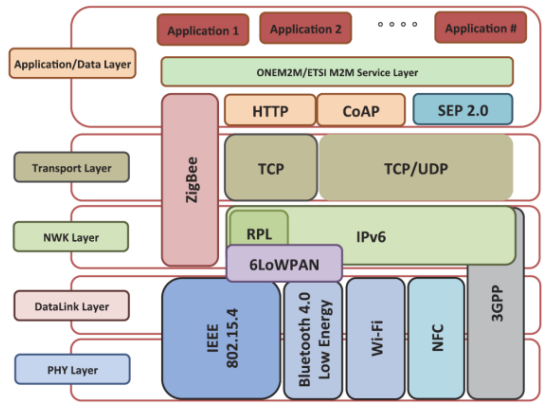

Fig. 1. Relation of the IoT technologies and the OSI model

The radio interference created in the 2.4 GHz radio channels produces three times higher error rate for the IEEE 802.15.4 channel as for the WiFi. Development of stochastic models for systems distributed in space and time and their application in the description of radio channel noise characteristics in WiFi system with high number of base stations serving as sensor nodes [6]. Based on this idea a new kriging method is proposed for continuous extrapolation of the signal field intensity in 4D physical coordinates (space-time domains) not sampled by the discrete sensor nodes [7].

Clustering method was developed and applied to extract information content from sensor network data sets and application of it to characterize the resource usage of a supercomputer system. The method based on artificial neural networks, cluster analysis and wavelets reduces by one order of magnitude the number of variables needed to be sampled to presage

surprise events at the CEP (Complex Event Processing) and ESP (Event Stream Processing) supported services based on huge number of logical and physical sensor nodes [8].

3. Cyber-Physical Systems

A Cyber-Physical System is a special case of the Internet of Things. It is characterised by a very intense interaction with the physical processes, and usually cooperating nodes solve a common task. Within the frame of this project we aimed at combining the advantageous behaviour of embedded- and IT systems. We are going to extend the possibilities of embedded systems through utilization of high performance IT solutions and through the possibility of strong cooperation of separate nodes by means of interconnections through the high speed internet. However, in our view, the interconnection of large set of embedded systems serve as general purpose cyber-physical resources, rather than resources for dedicated purposes.

We envision a farm of embedded systems, with a large set of sensors and actuators as a universal infrastructure for gathering information from the physical world, for interacting with it through actuators, and also as a universal computation resource [9]. A user utilising this infrastructure can develop a new application, based on the available new and historic sensor information and can influence the environment (in a controlled way).

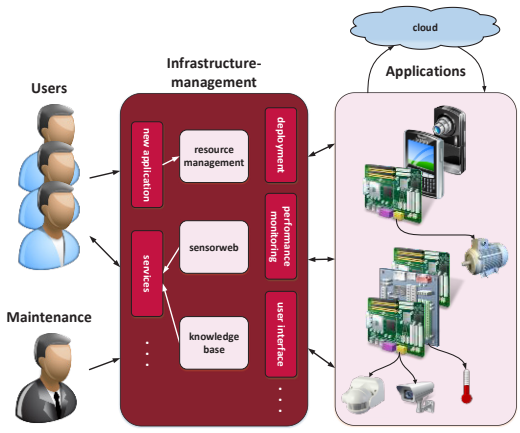

Fig. 2. Architecture of the Cyber-Physical infrastructure

New applications are automatically and dynamically allocated to embedded computing devices. Based on the measured resource utilization, the tasks are reallocated among devices in run-time by means of *Design-Space Exploration*. The above concept requires high level virtualization [10]. We use sensor virtualization (common interface, description structure and database) to access the information about physical processes from any embedded nodes. The possibility of reallocation of tasks also requires a certain level of virtualization, which might range from process *virtualization* to full platform virtualization. The computing nodes are strong enough to host virtual machines, guest operation systems and several applications at a time. However, if the application is very compute intensive, we can delegate certain parts of the calculations to the cloud (*cloud computing*) [11]. Sensor information collected by embedded nodes are accessible through an *ontology*, which allows the users to search for special types of sensors, or based on location, availability, accuracy etc.

4. Context-aware communication in the IoT

The 128-bit IPv6 addresses provide an unthinkably large address space, making it possible to assign trillions of addresses to each square centimetre on the surface of the Earth, so it is hard to envision any future scenario, including the "wildest" IoT-related predictions, whose needs would not be satisfied. However, in certain cases, size does not matter, or at least it is not the only thing that matters. The more relevant question is how can be those addresses used, how large will grow the routing tables, or how fast and how efficient can be the subsequent routing protocols and communication schemes. In the IoT we will probably very rarely use individual IPv6 addresses as is, we will not address a given sensor individually, but rather a group of smart "things" having in common some context-related characteristics. Therefore we propose to use a context-aware addressing and routing scheme, in which the network routes the queries to the proper place(s) based on a set of context parameters, but without knowing the IP addresses of the concerned objects.

We propose to encode context parameters in Bloom filters, which are considered a very resource-efficient and easy-to-process solution to handle set operations. IoT nodes will probably be grouped together in smaller areas behind several edge nodes connecting them to the traditional Internet architecture. The devices behind a specific edge will build and maintain a multi-hop tree over which context information in Bloom-filters can be easily exchanged and aggregated. When a context-based query is initiated, it will be rapidly routed to areas where IoT nodes exist, conforming to the requested context. The basic idea of this context-aware addressing solution was described in [12]. Currently we are working on implementing this approach in an IoT simulator and analysing its efficiency in different setups.

However, context-information can be very complex, involving several temporal and spatial correlations between the different context parameters. Capturing the evolution of most of these parameters is important, but usually only a very reduced set of these parameters affect effectively the behaviour of a given device, application or person. Another aspect of our research was therefore to provide a solution for filtering out these parameters based on the Hierarchical Temporal Memory approach (HTM), as described in [13].

5. E-health powered by IoT

In the present world, millions of people die every year due to lack of information about their health. Increased costs in the healthcare system could be reduced, if it would give more attention to disease prevention through regular assessment of health status and their treatment in the early stages. Our research is oriented to develop technologies for independent daily life assistance of elderly persons or others with disabilities and to improve the quality of human life using Internet of things (IoT) techniques.

Our scope is to bring together latest achievements in the domains of IoT and of assistive technologies in order to develop a complex assistive system with adaptive capability. Learning behaviour that allows living for as long as possible in familiar environment is also in focus of our research work. We use IoT technologies to monitor in real time the state of a patient or to get sensitive data in order to subsequently analyse and to enhance the medical diagnosis.

We have developed an assistive assembly consisting of a smart and assistive environment. This equipment allows also indoor localization based on wireless sensor network and Wi-Fi infrastructure [14]. It was developed a human activity and health monitoring system [15], an assistive and telepresence robot, together with the related components and cloud services. For activity and health pattern recognition we developed a hardware module for vital parameters monitoring (temperature, heart rate, acceleration). The acquired data is used to train neural network that allows recognition of activity or health status of the patient and trigger alert signal in case of unusual state detection. We implemented and tested a recognition system for arm posture, body postures and simple activities like standing, sitting, walking, running, etc., see Fig. 3. These states and movement forms were correlated with the data acquired from a heart rate sensor. The recognition rate of the body postures was over 99 % on the data sets used for training.

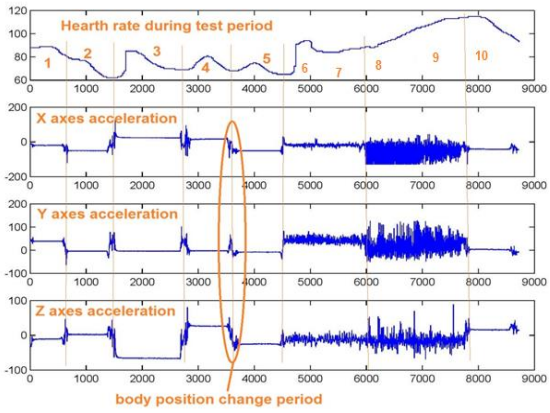

Fig. 3. Data acquired for 10 activities to be recognized.

On Fig. 4 can be seen that by setting the right threshold (red line) for the standard deviation, the static – dynamic postures discrimination can be easily differentiated. For recognizing the walking and running activities, we have extracted further relevant features from raw data set. The FFT transform was used to determine the stepping rate of a person as the most dominating frequency in the acceleration signal's spectrum.

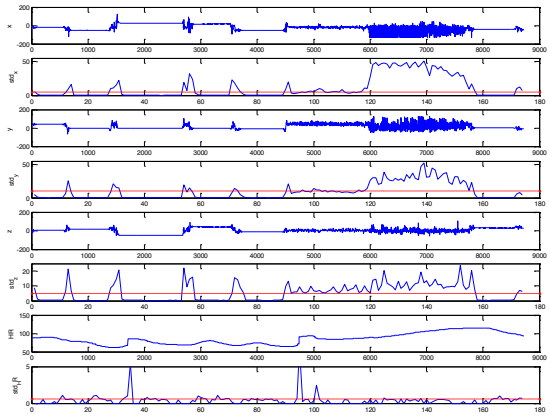

Fig. 4. Differentiating dynamic activity from static.

It was developed an own simulator application based on Qt application framework for feed forward neural networks [16]. Based on neural networks simulated in Matlab environment FPGA circuit was developed.

In our future development this monitoring module will be extended by new sensors (ECG, EMG, breath, etc.) and rules for sensor data fusion and fuzzy logic will be applied to enhance the body activity recognition. Further research based on different types of ANN is needed to simulate other activity/health status recognition. The best performing ANN type will be used to implement new recognition module based on FPGAs.

6. Weather prediction systems and analysis

The main area of our research is statistical post-processing of ensemble forecasts which is a pioneering work in this direction in Hungary. A forecast ensemble is obtained from several runs of a numerical weather prediction model with different initial conditions and makes possible the estimation of the probability distribution of future weather variables. This allows probabilistic weather forecasting, where not only the future atmospheric states are predicted, but also the related uncertainty information [20]. Recently several meteorological services provide ensemble forecasts, the leading organization is the ECMWF, while the Hungarian Meteorological Service (HMS) operates the ALADIN-HUNEPS ensemble prediction system. However, the spread of these forecast ensembles is often too small, they are uncalibrated and statistical methods are needed to account for this deficiency. The most popular tools of post-processing are the Bayesian Model Averaging (BMA) [22] and the Ensemble

Model Output Statistics (EMOS) [21]. Both approaches provide estimates of the densities of the predictable weather quantities and once a predictive density is given, a point forecast can be easily determined (e.g., mean or median value). As a first step we tested the existing BMA models implemented in the R package ensembleBMA on ALADIN-HUNEPS ensemble forecasts of wind speed [18] and temperature [19]. We found that statistical post-processing significantly improves the calibration of probabilistic and accuracy of point forecasts. We also developed a new univariate BMA model for wind speed prediction [17] and a bivariate BMA model for joint calibration of ensemble forecasts of wind speed and temperature. Both methods were successfully tested on ALADIN-HUNEPS ensemble forecasts and on forecasts of the University of Washington Mesoscale Ensemble and the results were compared to the predictive performances of the existing methods. We also performed a detailed comparison of BMA and EMOS calibration of ALADIN-HUNEPS temperature and wind speed forecasts and recently we are working on a new EMOS model for wind speed prediction. The predictive performance of this new model has already been tested on forecasts of wind speed of the UWME and of the ECMWF and ALADIN-HUNEPS ensemble prediction systems.
All new models are implemented in R and compatible with the existing ensembleBMA and ensembleMOS packages. The final goal of our research is the operation application of some statistical post-processing methods at the HMS.

7. Virtual service platforms and testbeds

More than 25 years of continuous development in the research networking area and later in the areas of those higher level e-Infrastructure services as grids, clouds, HPC, storage, collaboration and data infrastructures, have resulted in a leading edge e-Infrastructure system in Hungary that offers the provision of national and international services for the entire Hungarian research and education as well as public collection communities. The service portfolio includes, among others, communication, information access, and collaboration tools and platforms (e.g. remote co-operation and virtual community environments). The country-wide Hungarian e-Infrastructure is connected into the European and global e-Infrastructures via GÉANT, the European backbone of the research and education community. The services, having been developed and being operated by the NIIF Institute, are available also for the Future Internet research communities, and are extended to novel opportunities such as providing Virtual Research Environment (VRE) platforms and supporting Virtual Research Organisations (VRO) by making applications VO ready. An important special example of the major activities related to the e-Infrastructure is the development of a Shibboleth 2.x IdP X.509/LDAP authentication module. The basic motivation is to provide the opportunity of using hardware tokens as authentication source. SPs can decide if they want to force the X.509 authentication, or intend to simply keep a password based solution. Besides Shibboleth X.509 authentication (with or without PKI), also X.509 + LDAP certificate authentication and combining X.509 with username/password authentication are also possible options.
Based on GÉANT, also a specific, reconfigurable testbed operating in a federated virtual networking environment is provided by NIIFI, and its European partners, to the R&D community. The Hungarian segment of the testbed infrastructure is built on the high speed network of NIIFI and, together with its international connections, it is also available for supporting Future Internet research activities. Application of a two-factor authentication module for simpleSAMLphp in the federated virtual networking environment and in the testbed system has been developed, in order to achieve increased security by pairing a time-based token with other credentials, such as a username and a password. SimpleSAMLphp is used as a SAML2 Single-Sign-on solution based on php. Google Authenticator implements time-based one-time password (TOTP) security tokens from RFC6238 in mobile apps made by Google. The Authenticator provides a six digit one-time password users must provide in addition to their username and password to log into Google services. The Authenticator can also generate codes for third party applications, such as password managers or file hosting services.

8. Summary

The FIRST/IoT R&D project executed with the collaboration of several universities and institutes from Hungary, United Kingdom, Romania, Ukraine and Serbia has considerable effect on the Internet of Things topic. More than thirty journal and conference proceeding papers were published based on the theoretical and practical research work during the last two years. The results obtained in this way are considered promising basics for the continuation of the IoT field by next international joint projects.

Acknowledgement

The publication/poster was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund. Tamás Dabóczi also acknowledges the support of ARTEMIS JU and the Hungarian Ministry of National Development (NFM) in frame of the R5-COP (Reconfigurable ROS-based Resilient Reasoning Robotic Cooperating Systems) project.