Volume 1, Issue 2, August 01, 2016

TC-CPS Newsletter

IEEE TC-CPS Cyber-Physical Systems

Editorial

Cyber-Physical Systems (CPS) are characterized by the strong interactions among cyber components and dynamic physical components. CPS system examples include automotive and transportation systems, smart home, building and community, smart battery and energy systems, surveillance systems, cyber-physical biochip, and wearable devices. Due to the deeply complex intertwining among different components, CPS designs pose fundamental challenges in multiple aspects such as performance, energy, security, reliability, fault tolerance and flexibility. Innovative design techniques, algorithms and tools addressing the unique CPS challenges, such as the fast increase of system scale and complexity, the close interactions with dynamic physical environment and human activities, the significant uncertainties in sensor readings, the employment of distributed architectural platforms, and the tight real-time constraints, are highly desirable.

The IEEE TC-CPS Newsletter, published twice a year, aims to report the recent advances on technologies, educations and opportunities and, consequently, grow the research and education activities in this area. This letter is affiliated with the Technical Committee on Cybernetics for Cyber Physical Systems under the IEEE Systems, Man, and Cybernetics Society. TC-CPS aims at promoting interdisciplinary research and education in the field of CPS.

This issue of the newsletter showcases the state-of-the-art developments covering several emerging areas: machine monitoring, automobile, social cloud, energy, etc. Professional articles are solicited from technical experts to provide an in-depth review of these areas. These articles can be found in the section of “Technical Articles”. In the section of “Technical Activities”, recent activities organized by the TC-CPS, including workshops, special issues, etc., are summarized. Finally, the Call for Contributions can be found at the end of this issue to solicit high-quality submissions.

I would like to express my great appreciation to all Associate Editors (Yier Jin, Rajiv Ranjan, Yiyu Shi, Bei Yu and Qi Zhu) for their dedicated effort and strong support in organizing this letter. I wish to thank all authors who have contributed their professional articles to this issue. Finally, please allow me to welcome all of you to the founding issue of the TC-CPS Newsletter. I hope that you will have an enjoyable moment when reading the letter!

Xin Li
TC-CPS Editor
Carnegie Mellon University

Luminescent Solar Concentrator-Based Reconfigurable Photovoltaic System for EV/HEV

Yanzhi Wang, Caiwen Ding, Syracuse University

Photovoltaic (PV) cells provide us a clean and quiet form of electrical energy generation, and can be an ideal power source for EVs and HEVs. In general, the onboard PV system can provide up to 20%-30% of propelling power for a normal EV/HEV during cruising and city driving (which takes <10kW), and perhaps more importantly, it could charge the EV/HEV battery pack during parking time to reduce the recharging requirement and mitigate the power demand from the grid.

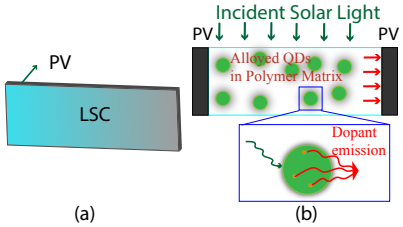

Figure 1: (a) Top view and (b) vertical cross section schematic of LSC-enhanced PV cell.

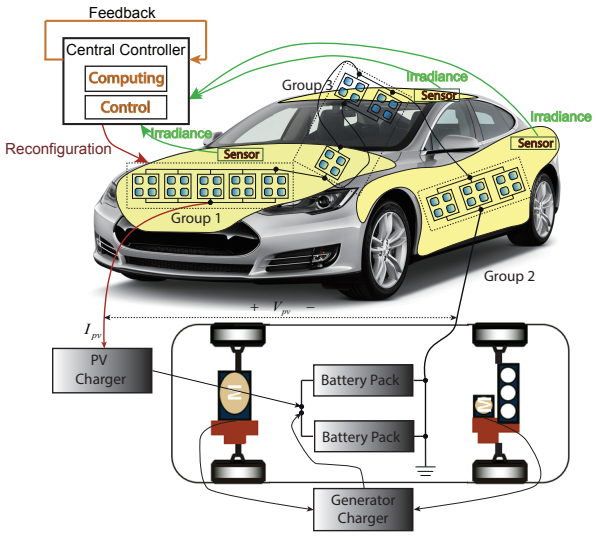

Figure 2: System diagram of an LSC-enhanced reconfigurable onboard PV system.

To increase PV power generation for EV/HEV, we should enlarge onboard PV cell modules by using all possible vehicle surface areas including the rooftop, hood, trunk, and door panels. These PV cell modules are connected to the EV/HEV battery pack through one power converter. This structure is called string charger architecture, and is a practical choice for onboard PV system accounting for cost considerations and high voltage of EV/HEV battery pack [1, 2].

However, onboard PV systems for EV/HEV exhibit certain limitations. Besides the relatively low energy conversion efficiency, common PV modules normally use flat-plate PV cells and may not fit the streamlined surface of trendy vehicles. PV cells (with typically a dark blue color) may not satisfy aesthetic standards of modern vehicles. Furthermore, the solar irradiance levels on different PV cells may be different from each other due to different solar incidence angles. For example, the solar irradiance level on the rooftop PV cells is higher compared with the door panel at noon due to smaller solar incidence angle. Under the non-uniform distribution of solar irradiance, it is difficult to make all PV cells operate at their maximum power points (MPPs) simultaneously [3], because the shaded PV cells will affect the operating point of lighted cells connected in series. This effect can lead to a dramatic output power degradation of PV system.

In order to address the limitation on appearance and compatibility with EV/HEV, we adopt semiconductor nano-materials-based *luminescent solar concentrator* (LSC)-enhanced PV cells for onboard PV systems. An LSC-enhanced PV cell (shown in Fig. 1) comprises an LSC polymer film [4] with vertically surrounding PV strips. The LSC polymer is magnetically doped by quantum dots (QDs), and can concentrate both direct sunlight and diffuse light onto attached PV strips to allow them to operate at higher efficiency. This new technology could mitigate the above limitations because (i) LSC-enhanced PV cells are flexible and can fit the surface streamlined designs of modern vehicles. (ii) LSC polymers are thin and transparent, and thus they do not affect aesthetic requirements of vehicle designs. (iii) LSC can potentially enhance the overall output power and reduce capital cost.

In order to address the problem induced by non-uniform solar irradiances, we have proposed a dynamic PV array reconfiguration technique which can extract the maximum output power of all PV cells simultaneously, thereby achieving a transformative improvement in the output power of onboard PV system. The reconfiguration mechanism exhibits polynomial-time complexity, and changes the internal connections of PV cells in the array without changing their physical locations. The reconfiguration mechanism should be triggered frequently to track the changes on solar irradiances during vehicle driving. The proposed LSC-based reconfigurable PV system for EV/HEV could simultaneously achieve high and reliable output power (2.5X enhancement compared with onboard PV system without reconfiguration), low capital cost and timing/energy overheads (less than 1% energy overhead), and full compatibility with EV/HEV.

References

[1] W. Xiao, N. Ozog, and W. G. Dunford, "Topology study of photovoltaic interface for maximum power point tracking," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 3, pp. 1696–1704, 2007.

[2] C. Hamilton, G. Gamboa, J. Elmes, R. Kerley, A. Arias, M. Pepper, J. Shen, and I. Batarseh, "System architecture of a modular direct-dc pv charging station for plug-in electric vehicles," in *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*. IEEE, 2010, pp. 2516–2520.

[3] H. Patel and V. Agarwal, "Maximum power point tracking scheme for pv systems operating under partially shaded conditions," *IEEE transactions on industrial electronics*, vol. 55, no. 4, pp. 1689–1698, 2008.

[4] F. Meinardi, A. Colombo, K. A. Velizhanin, R. Simonutti, M. Lorenzon, L. Beverina, R. Viswanatha, V. I. Klimov, and S. Brovelli, "Large-area luminescent solar concentrators based on/stokes-shift-engineered/'nanocrystals in a mass-polymerized pmma matrix," *Nature Photonics*, vol. 8, no. 5, pp. 392–399, 2014.

Accumulative Computing: Sensing With Unlimited Free Energy

Jingtong Hu, Oklahoma State University

1 Introduction and Motivation

Sensors are an integral part of Cyber-Physical Systems (CPS). While battery and cable power are still the major energy source for many sensors, there is a class of devices in which it is challenging to employ battery or cable power since it is inconvenient, costly or even dangerous to replace or service them. Examples of such applications include implantable sensor, wearable health monitor, water pipeline or building HVAC status monitor, soil or water pollution monitor, etc. Energy harvesting techniques, which generate electric energy from their ambient environment using direct energy conversion techniques, are very attractive to these applications because they can eliminate the need for batteries or wires and enable long-term adoption of these systems.

Figure 1 shows the architecture for a typical energy harvesting based sensing system. Ambient energy such as light, kinetic, RF, thermal, or even biochemical energy, are harvested and stored in a small capacitor, which can be used to power the processor and peripheral devices with on-chip converters [6]. However, there is an intrinsic drawback with harvested energy sources. They are **intermittent**. Since almost all traditional computer systems are designed based on the assumption of a stable power supply, none of them can make significant progress under frequently interrupted power. In order to take advantage of unlimited free energy supply, a new computing paradigm which can make progress even under intermittent power is needed.

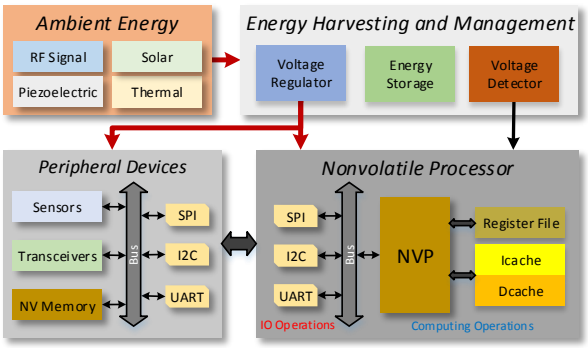

Figure 1: Energy Harvesting System

In order to make progress, we have to **accumulate** the computing across intermittent power cycles. The key idea is to save the processor's volatile registers to a non-volatile memory (NVM) when there is a power failure and restore the processor state when the power comes back on. There have been several works to achieve this with either software assisted approach [1, 2, 5] or hardware approach [4]. While existing research shows exciting advancement, there are still challenges that need to be answered to make self-powered accumulative computing a mature platform.

- First, while most existing works are successful in achieving the continuous computing functionality, few of them considered optimizing the checkpointing efficiency. On one hand, the energy harvested in such systems is usually limited. On the other hand, not only registers need to be checkpointed, but on-chip and off-chip memories also need to be checkpointed if they are volatile. Therefore, fast and efficient checkpointing is needed for the whole volatile memory hierarchy to ensure successful checkpointing. Meanwhile, more energy can be used for system forward progress.
- Second, while the computing status can be saved, I/O interfaces associated with peripheral sensors and communication devices are hard to checkpoint due to their time-sensitivity and atomicity. In many cases, interrupted operations need to be restarted from the beginning, which will severely affect the forward progress. Meanwhile, checkpointing for processor and I/O operations in interrupt service routines (ISRs) has to be handled properly to ensure correct execution.
- Third, when multiple tasks are running concurrently in the system, the OS scheduler and task management will also affect the forward progress upon power failure.
- Additionally, our study [7] showed that without considering the volatility across the memory hierarchy, data inconsistency might happen and lead to fatal errors.

2 Checkpoint Efficiency Optimization

Several works have been done to optimize the checkpoint efficiency. First, we have developed a stack trimming technique to minimize stack data that need to be checkpointed [3]. The idea is to reduce the stack size via address space sharing. Within a program, each function instance is associated with a *frame* (also called *active record*) to store the context information for this function. Local data, including local variables and compilation temporary variables, are stored in this frame. A conventional stack based allocator works as follows: a specific memory address is assigned to the *main* function's frame. When a function is called, the callee function's frame is allocated on top of the caller function's frame. When a function returns, the callee function's frame is deallocated from the top of caller function's frame. Traditionally, the stack space is separately allocated for the caller and callee functions, which is conservative and results in a large stack size.

```
 1 struct T
 2 {
 3 int i;
 4 int j;
 5 char arr[10];
 6 };

 7 int cpyT( T *t1, *t2 )
 8 {
 9 int a;
10 int b;
11 modify(&a);
12 t1.i = t2.i + a;
13 modify(&b);
14 t1.j = t2.j + b;
15 strcpy(t1.arr, t2.arr);
16 return 0;
17 }
```

Figure 2: An example program

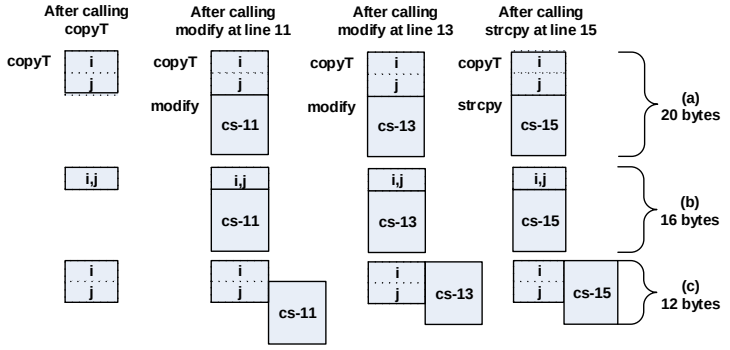

Figure 3: Comparison of stack size under different stack allocation schemes. Assume that the frame size of *copyT*, *modify* and *strcpy* is 8, 4 and 12 bytes, respectively.

A simple motivation example is presented to illustrate how stack allocation schemes affect the stack size. The example code is shown in Figure 2. Note that each call site (cs) is also viewed as a local object, and its size is equal to the callee function's frame size. Figure 3 illustrates the stack size under different schemes. Figure 3(a) shows that the conventional allocation scheme, without any overlay in stack, holds the largest stack size of 20 bytes. Since *i* and *j* have disjoint live ranges, they can be coalesced. The result is shown in Figure 3(b). Here, the frame size of *copyT* is reduced by 4 bytes, and the maximum stack size is reduced to 16 bytes. In order to further reduce stack size, we aggressively overlay call sites with disjoint live ranges [3]. Figure 3(c) shows the result, in which the maximum stack size can be reduced to 12 bytes. From this example, we can see that objects with disjoint live ranges can share the same address without violating the data integrity and thus reduce the stack size. The experimental results show that the proposed technique can reduce the stack size by 28.6% on average for a wide range of benchmarks.

In addition to the stack trimming, there are also optimization opportunities in the temporal domain. Figure 4(a) shows an example code, where the *main* function invokes function *g*; *g* invokes *h*; and *h* invokes *i*. The stack usage is shown in Fig. 4(b). From the figure, we can see that the stack size fluctuates as the functions are invoked and return. Assume that the system detects power failure at time t_1. The conventional backup strategy is *instant backup*, where all the processor states are backed up immediately at t_1. In this case, this system needs to checkpoint four stack frames. However, instead of consuming a large portion of remaining energy to checkpoint, we can spend some energy to continue the program execution until t_2. At t_2, there is only one stack frame to checkpoint since all the callees already returned. Based on this observation, we developed a three-step approach [8], in which the best backup positions are derived in polynomial time. The evaluation results show considerable checkpoint content

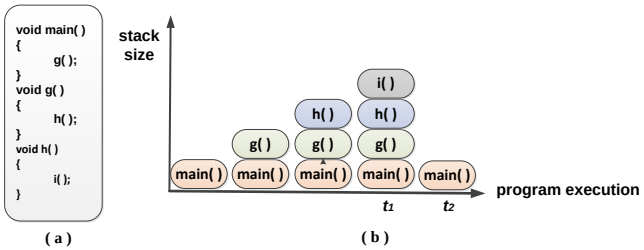

Figure 4: Stack Fluctuation.

reduction compared with instant checkpoint.

3 Conclusion

Realizing accumulative computing on unstable harvested energy will enable a new class of self-powered sensing/monitoring systems that can last for years and require the least maintenance effort in various non-timing critical applications. It will simplify system installation and maintenance in many areas such as health care, building monitoring and maintenance, traffic, agriculture and environment monitoring, and even crisis management. Meanwhile, it will help bridge the gap between ever-increasing electronic power needs and battery scalability and have the potential to provide a large infrastructure for opportunistic computing with great social impact. However, there all still several challenges to be answered to achieve the goal. This article presents two checkpoint efficiency optimization techniques which aim to overcome these challenges.

References

[1] D. Balsamo, A. S. Weddell, G. V. Merrett, B. M. Al-Hashimi, D. Brunelli, and L. Benini. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters*, 7(1):15–18, March 2015.

[2] Hrishikesh Jayakumar, Arnab Raha, Woo Suk Lee, and Vijay Raghunathan. Quickrecall: A hw/sw approach for computing across power cycles in transiently powered computers. *J. Emerg. Technol. Comput. Syst.*, 12(1):8:1–8:19, August 2015.

[3] Qingan Li, Mengying Zhao, Jingtong Hu, Yongpan Liu, Yanxiang He, and Chun Jason Xue. Compiler directed automatic stack trimming for efficient non-volatile processors. In *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, pages 183:1–183:6, 2015.

[4] K. Ma, Y. Zheng, S. Li, K. Swaminathan, X. Li, Y. Liu, J. Sampson, Y. Xie, and V. Narayanan. Architecture exploration for ambient energy harvesting nonvolatile processors. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 526–537, Feb 2015.

[5] Benjamin Ransford, Jacob Sorber, and Kevin Fu. Mementos: System support for long-running computation on rfid-scale devices. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVI, pages 159–170, 2011.

[6] Umamaheswara Rao Tida, Cheng Zhuo, and Yiyu Shi. Novel through-silicon-via inductor-based on-chip dc-dc converter designs in 3d ics. *J. Emerg. Technol. Comput. Syst.*, 11(2):16:1–16:14, November 2014.

[7] Mimi Xie, Mengying Zhao, Chen Pan, Jingtong Hu, Yongpan Liu, and Chun Jason Xue. Fixing the broken time machine: consistency-aware checkpointing for energy harvesting powered non-volatile processor. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 184:1–184:6, 2015.

[8] Mengying Zhao, Qingan Li, Mimi Xie, Yongpan Liu, Jingtong Hu, and Chun Jason Xue. Software assisted non-volatile register reduction for energy harvesting based cyber-physical system. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE, pages 567–572, 2015.

End-to-End QoS Specification and Monitoring in the Internet of Things

Awatif Alqahtani[1], Ellis Solaiman[1], Rajkumar Buyya[2], Rajiv Ranjan[1]
[1]School of Computing Science, Newcastle University, United Kingdom
[2]Computing and Information Systems, University of Melbourne, Australia

The Internet of Things (IoT) is a computing paradigm where uniquely addressable objects such as Radio-Frequency IDentification (RFID) tags, sensors, actuators, and mobile phones, are able to connect via the Internet, and cooperate with each other in order to achieve common goals [1, 4]. The number of connected smart objects is estimated to reach 212 billion by the end of 2020 [2, 3]. Such large numbers of connected smart objects will generate huge volumes of data that needs to be analysed and stored [6]. According to a study conducted by IBM, 2.5 quintillion bytes of data is generated every day [7]. Storing and processing such large volumes of Big Data is non-trivial, and requires the flexibility offered by Cloud Computing [8]. Cloud Computing offers a pool of configurable resources (hardware/software) that are available on demand [9], allowing users to submit jobs to service providers on the basis of pay-per-use. While the IoT provides smart devices with the ability to sense and generate large amount of data that reflect the physical world in different forms and different data speeds, Cloud Computing offers advanced technologies for ingesting, analysing and storing data [10]. The number of applications based on IoT and Cloud Computing is projected to increase rapidly over next few years. To this end, Gascon and Asin [11] predicted that in near future there will be about 54 types of IoT applications for addressing different domain [5] specific problems: security and emergency, smart environment, smart cities, smart metering, smart water, smart animal farming, smart agriculture, industrial control, retail, logistics, domestic and home automation and e-Health.

1 Quality of Service roadmap for IoT applications

Expectation from services provided by the Internet of Things are no different from most traditional computer and Internet based services in that they must be delivered with guaranteed levels of quality of service (QoS). For example, in emergency response (ER) IoT application, there is a need to receive and analyse data from deployed sensors immediately and accurately in order to allow for timely response to potential damage that can be caused in natural disaster situations such as earthquakes, floods, and tsunamis [12]. Such IoT applications can be extremely time sensitive; any delay in the collection/transferring/ingestion/analysis of sensor data may have disastrous consequences.

As we note in our previous papers [4, 13], engineering IoT applications that can guarantee QoS is a challenging and not feasible with the current state-of-the-art available in context of IoT programming models (e.g. Amazon IoT, Google Cloud Dataflow, IBM Quark) and resource management methods [4, 33]. An important difficulty is that IoT application eco-systems are typically consists of several layers involving multiple, heterogeneous hardware and software resources; and data types from digital and human sensors. An example of an IoT eco-system is depicted in Figure 1, which consists of following programming and resource management layers; sensing layer, gateway layer, network layer, and cloud layer. Providing customers with QoS guarantees requires the technical ability to ensure that their QoS requirements will be observed across each of the layers of an IoT application eco-system.

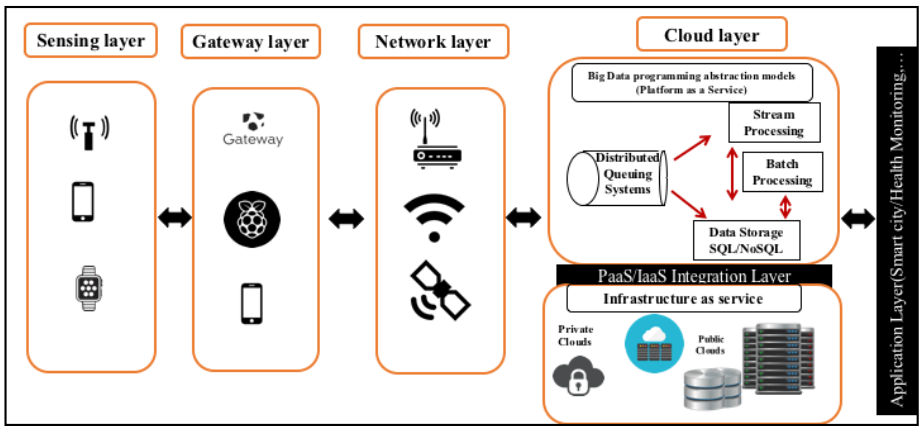

Figure 1: A multi-layered architecture IoT application eco-system involving Sensing, Gateway, Network, and Cloud layers.

2 Specification of IoT application specific QoS requirements within Service Level Agreements

In purely business context, QoS requirements are formally specified in a Service Level Agreement (SLA) document [3] which serves as the basis of legal agreement and understanding of service terms, conditions, and commitments between consumers and providers. For example, Amazon Web Services' SLA document stating the terms, conditions, and commitments for its S3 and EC2 services can be found at [34] and [35] respectively.

As IoT applications have layered architecture and complex Big Data flows across layers, there is a need to first model SLA for individual layers followed by their holistic aggregation. Such aggregated SLA document (template) will form basis for specifying an end-to-end SLA that can be used to specify the service terms, conditions and commitments for an IoT application. Notably, cross-layer SLAs in IoT have a strong dependency relationships with each of its upstream and downstream layers, regardless of whether this component is data, computing hardware, IoT sensor, software, or human. Thus violation of one or more constraints by one or more components (s) affects the adherence to the related SLA's terms.

To illustrate this concept, consider a remote health monitoring IoT application [13] where patients wear sensors and accelerometers to measure their heart rate and sugar levels, reminding them of the time to take medications, and detecting abnormal activities such as falling down. Subscribed patients might ask for a service that can satisfy the following high-level, strict SLA requirement: detecting abnormal activity, such as falling down, within x milliseconds, then alerting/notifying the ambulance, caregivers, and doctors within y minutes. To achieve this high-level SLA requirement, many nested-dependent QoS metrics should be considered, such as high-quality sensors with minimum event detection delay (within x milliseconds), available networks with low latency, and a high-alert detection and notification analytic service to deliver the desirable alerts to relevant healthcare providers and relatives. As patients need to receive the required emergency treatment based on their health status within y minutes, this means that the aggregation of the response time from each layer should be within the time constraints, i.e. less than or equal to y minutes. A delay in the network, for example, would lead to a late response at the alert generation front-end, which could exceed the time the patient and healthcare provider was expecting (y minutes). Specifying SLA requirements with their required level of QoS and monitoring their adherence to these specifications is a non-trivial task and includes many challenges such as:

A. Heterogeneity of Big Data sources and their distributed locations.

B. Heterogeneity of the key QoS metrics across layers.

C. Heterogeneity of application requirements.

D. Lack of unified/standard methods for collecting the required metrics across-layer and from multiple providers for end-to-end SLA monitoring purposes.

3 SLA specification and monitoring current research efforts:

Substantial research on the specification and monitoring of QoS and SLAs has been conducted for computer networks, web services, Grids, and Cloud Computing. But limited literature is available that deal the problem of specifying and monitoring end-to-end QoS and SLAs in an IoT application eco-system. For example, NetloggerÂăprovides an API that can be used by applications to check the load on network resources before and after performing operations / sending requests. However, Netlogger only monitors network resources and does not extend to other components of an IoT application [14, 15]. The Web Service Level Agreement (WSLA) standard described in [16] was developed for web service SLA specification. Also, WS-Agreement from the Open Grid Forum (OGF) defines a web service agreement specification as a protocol for launching an agreement between two parties. An illustration of how cloud providers in industry apply SLAs is shown in [21]. Cloud providers such as AmazonEC2, S3 (IaaS provider) and Windows Azure Compute and Storage, serve a pre-defined SLA, and the user can then choose the most appropriate provider that will fit their requirements. After entering into a contract with the selected provider, the SLA can be monitored against violations using third parties such as Cloudwatch, Cloudstatus and Monitis. The LoM2HiS Framework [14] aims to monitor and enforce SLA objectives in the cloud environment, especially; scalability, efficiency and reliability requirements. The framework aims to map low-level resource metrics to high SLAs objectives. However, the LoM2HiS Framework does not extend beyond the Cloud infrastructure layer. A European Commission Report on Cloud Computing Service Level Agreements [24] identifies and describes several interesting research efforts. SLA(T) by the SLA@SOI project [25, 27] is a model and language for service description that expresses the dependencies among services within / across layers in the Cloud. Another project (CONTRAIL) provides a quality model [28] for capturing different parameters of interest for customers and providers. The IRMOS project [3] proposes two SLAs at different levels: an application SLA to express high-level application terms between consumers and providers, and technical SLAs to express the low level QoS parameters linked to the infrastructure resources. Cloud4SOA [31], is a project which provides a unified monitoring interface that gives an overview of all of the customer deployments at one time, as well as selecting a set of unified metrics for monitoring both the execution and the usage of an application. IRMOS [32] provides an adaptable monitoring framework that collects data from both the application and technical level to monitor real-time application execution at time intervals based on the collected monitoring information and its associated SLA terms.

Despite a number of impressive research efforts into the specification and monitoring of QoS requirements within networks, web services, grids, and clouds, none of these are suitable in context of IoT applications. Developing formal approaches for the specification of QoS requirements and monitoring end-to-end IoT ecosystems is what we term as the next "grand challenge" for distributed systems researchers, and current platforms and techniques for monitoring IoT and Cloud computing fall short of this grand challenge.

References

[1] A. Flammini, and E. Sisinni. Wireless Sensor Networking in the Internet of Things and Cloud Computing Era. *Procedia Engineering*, vol. 87, pp. 672–679, Dec. 2014.

[2] J. Gantz and D. Reinsel. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the Far East. *IDC iView: IDC Anal. Future*, vol. 2007, pp.1–16, Dec. 2012.

[3] A. Galati et al. A WS-Agreement based SLA implementation for the CMAC platform. In *Economics of Grids, Clouds, Systems, and Services*, Springer-Verlag Heidelberg: Springer International Publishing, 2014.

[4] R. Ranjan. Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*, vol. 1, no. 1, pp.78–83, May 2014.

[5] R. Buyya, and A.V. Dastjerdi, eds. Internet of Things: Principles and Paradigms, Elsevier, 2016.

[6] K. Radha et al. Service Level Agreements in Cloud Computing and Big Data. *International Journal of Electrical and Computer Engineering*, 5(1), p.158, Feb. 2015.

[7] IBM What is Big Data? – Bringing Big Data to the Enterprise. https://www-01.ibm.com/software/data/bigdata/what-is-big-data.html

[8] Zheng, X., Martin, P., Brohman, K. and Da Xu, L. Cloud service negotiation in internet of things environment: a mixed approach. *Industrial Informatics, IEEE Transactions on*, vol. 10, pp. 1506-1515, May 2014.

[9] M. Díaz, C. Martín, and B. Rubio. State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing. *Journal of Network and Computer Applications*, 2016.

[10] M. Chen, S. Mao, and Y. Liu. Big data: a survey. *Mobile Networks and Applications*, vol. 19, no. 2, pp.171–209, Apr. 2014.

[11] D. Gascon, and A. Asin. 50 sensor applications for a smarter world. http://www.libelium.com/resources/top_50_iot_sensor_applications_ranking, 2015

[12] N. Li et al. A new methodology to support group decision-making for IoT-based emergency response systems. *Information Systems Frontiers*, vol.16, no.5, pp.953–977, 2014.

[13] P.P. Jayaraman et al. Orchestrating Quality of Service in the Cloud of Things Ecosystem. InÂă*IEEE International Symposium on Nanoelectronic and Information Systems*,Âăpp.185–190, December 2015.

[14] V.C. Emeakaroha et al. Low level metrics to high level SLAs-LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments. In *International Conference High Performance Computing and Simulation (HPCS)*, pp. 48–54), June 2010.

[15] D. Gunter et al. Netlogger: a toolkit for distributed system performance analysis. *International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pp. 267–273, 2000.

[16] A. Keller, and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, vol. 11, no. 1, pp.57–81, Mar 2003.

[17] M. Alhamad, T. Dillon, and E. Chang. Service level agreement for distributed services: a review. In *IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 1051–1054), Dec. 2011.

[18] A. Andrieux et al. Web services agreement specification (WS-Agreement). In *Open Grid Forum*, vol. 128, no. 1, p. 216, Mar, 2007.

[19] A. Sahai et al. Specifying and monitoring guarantees in commercial grids through SLA. In *Cluster Computing and the Grid*, 2003.

[20] M. Alhamad, T. Dillon, and E. Chang. A survey on SLA and performance measurement in cloud computing. In *On the Move to Meaningful Internet Systems: OTM*, pp. 469–477), Springer Berlin Heidelberg, 2011.

[21] L. Wu, and R. Buyya. Service Level Agreement (SLA) in utility computing systems. *IGI Global*, Apr. 2012.

[22] K. Alhamazani et al. Cross-Layer Multi-Cloud Real-Time Application QoS Monitoring and Benchmarking As-a-Service Framework. 2015.

[23] G. Cicotti et al. How to monitor QoS in cloud infrastructures: The QoSMONaaS approach. In *Intelligent Distributed Computing VI*, pp. 253–262), Springer Berlin Heidelberg, 2013.

[24] D. Kyriazis. Cloud computing service level agreements: exploitation of research results. *European Commission Directorate General Communications Networks Content and Technology Unit*, Tech. Rep, 2013.

[25] SLA@SOI Project. http://sla-at-soi.eu/

[26] RESERVOIR Project. http://www.reservoir-fp7.eu/

[27] 4CaaSt Project. http://4caast.morfeo-project.org/

[28] R. Cascella et al. Contrail: Distributed Application Deployment under SLA in Federated Heterogeneous Clouds. Springer, Lecture Notes in Computer Science, 2013.

[29] Q-ImPrESS Consortium. Service Architecture Meta-Model (SAMM). Project deliverable d2.1, Sep. 2008. Available: http://www.q-impress.eu/wordpress/wp-content/uploads/2009/05/d21-service_architecture_meta-model.pdf

[30] http://irmosproject.eu/Files/IRMOS_NEXOF-RA_SLAs_QoS.pdf

[31] Cloud4SOA Project. http://www.cloud4soa.eu/

[32] IRMOS Project. http://www.irmosproject.eu/

[33] L. Wang and R. Ranjan. Processing Distributed Internet of Things Data in Clouds. *IEEE Cloud Computing*, vol. 1, no.2, 2015.

[34] S3 SLA, Amazon Web Services. https://aws.amazon.com/s3/sla/

[35] EC2 SLA, Amazon Web Services. https://aws.amazon.com/ec2/sla/

Timing-Centric Software Synthesis for Cyber-Physical Systems

Qi Zhu, University of California, Riverside

Software design and implementation have become increasingly challenging for cyber-physical systems, with growing software complexity in terms of both scale and features as well as adoption of more distributed and networked hardware platforms. As an example, in automotive domain, embedded software increased from 2% to 13% of a vehicle's total value from year 2000 to 2010, and the number of lines of code increased from 1 million to more than 10 million [15, 1]. On the hardware side, the number of ECUs (electronic control units) in a standard car has gone from 20 to over 50 in the past decade [1]. The traditional federated architecture, where each function is deployed to one ECU and provided as a black-box by Tier-1 supplier, is shifting to the integrated architecture, in which one function can be distributed over multiple ECUs and multiple functions can be supported by one ECU [6]. This leads to significantly more sharing and contention among software functions over multicore and distributed platforms.

At the core of CPS software challenges is *timing*, which has critical impacts on both functional correctness and various design metrics such as control performance, fault tolerance and security [8, 16, 14]. In particular, the synthesis of CPS software remains hindered by timing-related issues: 1) *diversity of timing requirements* from different design metrics, some with conflicting constraints; 2) *complexity of timing analysis* under complex scale, hierarchy and concurrency of computation and communication; and 3) *uncertainty of timing behavior* resulting from dynamic environment, data input and platform conditions.

Current synthesis solutions and practices do not adequately address these timing challenges. Timing constraints are often set in an ad-hoc fashion without quantitative analysis of their impacts on multiple related metrics, and software synthesis is often conducted without continuous and holistic consideration of timing. In the widely-adopted model-based design paradigm, system functionality is first captured in a functional model for early simulation and

validation, and then commonly synthesized to software task implementations on hardware platforms. While timing is usually considered during the mapping of software tasks onto hardware platforms, it is *rarely addressed during the generation of software tasks from initial functional models*, and thereby leaving a significant gap in the synthesis process. As we have observed from our prior work [18, 3, 5, 19, 7, 4], such issues during software synthesis often lead to infeasible solutions, long design cycles, and ultimately inferior and error-prone CPS software implementations.

Our Software Synthesis Work: In past, we have worked on task mapping problems for distributed embedded systems and cyber-physical systems, including task allocation and scheduling for schedulability, latency, memory usages, and extensibility [21, 20, 22, 17], exploring task activation periods for schedulability [2], and task mapping with security considerations [9, 10, 11]. As traditional mapping problems, these approaches only focus on the mapping stage.

Recently, we started investigating timing-driven task generation, and have proposed algorithms for multi-task generation of finite state machines (FSMs) for timing robustness [19] and multi-task generation of dataflows with respect to schedulability, reusability and modularity [5, 3]. These works only address non-hierarchical functional models with single model of computation, however, the results have demonstrated *significant improvements from considering timing during task generation of functional models*. For instance, in [5], a 20%-40% reduction in latency is achieved by addressing timing during multi-task generation of dataflows. In [3], timing schedulability is addressed together with modularity (defined as the number of generated runnable functions [12, 13]) and reusability during synthesis of dataflows. It demonstrates that for a fuel injection system example, only considering modularity and reusability during synthesis (with algorithms from literature [12, 13]) results in infeasible solutions. While using our approach that considers timing during dataflow synthesis, a trade-off of modularity for schedulability results in multiple task generation solutions that can be feasibly allocated and scheduled onto the hardware platform.

Software synthesis for cyber-physical systems is a critical and challenging area. The works above have only addressed the tip of the iceberg. There is an urgent need to have more design automation methods and tools to tackle the challenges in CPS software design, implementation and validation.

References

[1] R. N. Charette. This Car Runs on Code. *IEEE Spectrum*, February 2009.

[2] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. Period Optimization for Hard Real-time Distributed Automotive Systems. In *Design Automation Conference (DAC'07)*, June 2007.

[3] P. Deng, F. Cremona, Q. Zhu, M. Di Natale, and H. Zeng. A Model-Based Synthesis Flow for Automotive CPS. In *Cyber-Physical Systems (ICCPS), 2015 ACM/IEEE International Conference on*, pages 198–207, April 2015.

[4] P. Deng, Q. Zhu, A. Davare, A. Mourikis, X. Liu, and M. Di Natale. An Efficient Control-Driven Period Optimization Algorithm for Distributed Real-Time Systems. *IEEE Transactions on Computers*, PP(99):1–1, 2016.

[5] P. Deng, Q. Zhu, M. Di Natale, and H. Zeng. Task Synthesis for Latency-Sensitive Synchronous Block Diagram. In *Industrial Embedded Systems (SIES), 2014 9th IEEE International Symposium on*, pages 112–121, June 2014.

[6] M. Di Natale and A. Sangiovanni-Vincentelli. Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools. *Proceedings of the IEEE*, 98(4):603 –620, april 2010.

[7] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. Sangiovanni-Vincentelli, and E. Lee. Metronomy: A Function-Architecture Co-Simulation Framework For Timing Verification Of Cyber-Physical Systems. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2014 International Conference on*, pages 1–10, Oct 2014.

[8] E. A. Lee and S. A. Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Lee & Seshia, 2011.

[9] C. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli. Security-Aware Mapping for CAN-Based Real-Time Distributed Automotive Systems. In *Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on*, pages 115–121, 2013.

[10] C. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. Security-Aware Mapping for TDMA-Based Real-Time Distributed Systems. In *Computer-Aided Design (ICCAD), 2014 IEEE/ACM International Conference on*, pages 24–31, Nov 2014.

[11] C.-W. Lin, B. Zheng, Q. Zhu, and A. Sangiovanni-Vincentelli. Security-Aware Design Methodology and Optimization for Automotive Systems. *ACM Trans. Des. Autom. Electron. Syst.*, 21(1):18:1–18:26, Dec. 2015.

[12] R. Lublinerman, C. Szegedy, and S. Tripakis. Modular Code Generation from Synchronous Block Diagrams: Modularity vs. Code Size. In *Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '09, pages 78–89, New York, NY, USA, 2009. ACM.

[13] R. Lublinerman and S. Tripakis. Modularity vs. Reusability: Code Generation from Synchronous Block Diagrams. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '08, pages 1504–1509, New York, NY, USA, 2008. ACM.

[14] F. Mueller. Challenges for Cyber-Physical Systems: Security, Timing Analysis and Soft Error Protection. In *High-Confidence Software Platforms for Cyber-Physical Systems (HCSP-CPS) Workshop, Alexandria, Virginia*, page 4, 2006.

[15] A. Sangiovanni-Vincentelli. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design. *Proceedings of the IEEE*, 95(3):467–506, March 2007.

[16] A. Sangiovanni-Vincentelli and M. Di Natale. Embedded System Design for Automotive Applications. *Computer*, 40(10):42 –51, 2007.

[17] H. Zeng, M. D. Natale, and Q. Zhu. Minimizing Stack and Communication Memory Usage in Real-Time Embedded Applications. *ACM Trans. Embed. Comput. Syst.*, 13(5s):149:1–149:25, July 2014.

[18] B. Zheng, P. Deng, R. Anguluri, Q. Zhu, and F. Pasqualetti. Cross-Layer Codesign for Secure Cyber-Physical Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):699–711, May 2016.

[19] Q. Zhu, P. Deng, M. Di Natale, and H. Zeng. Robust and Extensible Task Implementations of Synchronous Finite State Machines. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 1319–1324, March 2013.

[20] Q. Zhu, Y. Yang, M. D. Natale, E. Scholte, and A. Sangiovanni-Vincentelli. Optimizing the Software Architecture for Extensibility in Hard Real-Time Distributed Systems. *the IEEE Transactions on Industrial Informatics*, 6(4):621–636, 2010.

[21] Q. Zhu, Y. Yang, E. Scholte, M. D. Natale, and A. Sangiovanni-Vincentelli. Optimizing Extensibility in Hard Real-Time Distributed Systems. In *RTAS '09: Proceedings of the 2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 275–284, 2009.

[22] Q. Zhu, H. Zeng, W. Zheng, M. D. Natale, and A. Sangiovanni-Vincentelli. Optimization of Task Allocation and Priority Assignment in Gard Real-Time Distributed Systems. *ACM Trans. Embed. Comput. Syst.*, 11(4):85:1–85:30, 2012.

Summary of Activities (2016.01 – 2016.07)

1 Workshops

- DAC-2016 Workshop on Design Automation for Cyber-Physical Systems (CPSDA-2016)
- INFOCOM-2016 Workshop on Cross-Layer Cyber-Physical Systems (CPSS-2016)

2 Special Issues in Academic Journals

- IEEE Transactions on Sustainable Computing (TSUSC) Special Issue on Sustainable Cyber-Physical Systems
- IEEE Transactions on Big Data Special Issue on Big Data for Cyber-Physical Systems
- ACM Transactions on Cyber-Physical Systems (TCPS) Special Issue on Smart Homes, Buildings, and Infrastructure
- Integration, The VLSI Journal Special Session on Hardware Assisted Techniques for IoT and Big Data Applications
- IEEE Transactions on CAD Special Issue on CAD for Cyber-Physical System
- IEEE Transactions on Computers Special Issue on Smart City Computing
- IEEE Transactions on Multi-Scale Computing Systems Special Issue on Hardware Software Crosslayer Technologies for Trustworthy and Secure Computing

3 Special Sessions in Academic Conferences

- ISVLSI-2016 Special Session on Cyber-Physical Systems: Architecture and Security in Smart Buildings and Autonomous Driving
- ISVLSI-2016 Special Session on Emerging Devices for Hardware Security: Fiction or Future

4 Book Publications

- Springer Book "Leveraging Big Data Techniques for Cyber-Physical Systems"

Call for Contributions

Newsletter of Technical Committee on Cyber-Physical Systems (IEEE SMC Society)

The newsletter of Technical Committee on Cyber-Physical Systems (TC-CPS) aims to provide timely updates on technologies, educations and opportunities in the field of cyber-physical systems (CPS). The letter will be published twice a year: one issue in February and the other issue in October. We are soliciting contributions to the newsletter. Topics of interest include (but are not limited to):

- Embedded system design for CPS
- Real-time system design and scheduling for CPS
- Distributed computing and control for CPS
- Resilient and robust system design for CPS
- Security issues for CPS
- Formal methods for modeling and verification of CPS
- Emerging applications such as automotive system, smart energy system, internet of things, biomedical device, etc.

Please directly contact the editors and/or associate editors by email to submit your contributions.

Submission Deadline:

All contributions must be submitted by **Jan. 1st, 2017** in order to be included in the February issue of the newsletter.

Editors:

- Helen Li, University of Pittsburgh, USA, hal66@pitt.edu

Associate Editors:

- Yier Jin, University of Central Florida, USA, yier.jin@eecs.ucf.edu
- Rajiv Ranjan, Newcastle University, United Kingdom, raj.ranjan@ncl.ac.uk
- Yiyu Shi, University of Notre Dame, USA, yshi4@nd.edu
- Bei Yu, Chinese University of Hong Kong, Hong Kong, byu@cse.cuhk.edu.hk
- Qi Zhu, University of California at Riverside, USA, qzhu@ece.ucr.edu