Cognitive IoT Systems via Adaptive Swarm Intelligence

Augusto Vega and Pradip Bose
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA, 10598
Email: {ajvega,pbose}@us.ibm.com

***Abstract*—We present a novel paradigm called *Adaptive Swarm Intelligence* (ASI) where heterogeneous devices (or "agents") engage in collaborative "swarm" computing for robust and adaptive real-time operation. ASI, a paradigm inspired by the collaborative and decentralized behavior of some systems in nature, finds application in a myriad of scenarios, in domains like the IoT, mobile computing and distributed systems. Examples include network cybersecurity, connected/autonomous cars, and other types of unmanned vehicles, like "intelligent" drone swarms. This is by no means an exhaustive list but it gives an indication of the many and diverse domains that can benefit from this paradigm. This paper presents a specific ASI case study for cooperative sensor fusion in prospective connected/autonomous vehicles, which constitutes the driving application of the IBM-led "Efficient Programmability of Cognitive Heterogeneous Systems" (EPOCHS) project under the DARPA DSSoC program. Due to the magnitude of EPOCHS, we focus on one specific piece of our project: the *EPOCHS Reference Application* (ERA) for multi-vehicle sensor fusion. We show characterization results on a x86 system that allow us to draft preliminary conclusions about ERA's performance characteristics and real-time needs. The paper briefly describes EPOCHS' roadmap and future work.**

***Keywords*—embedded system; mobile cognition; swarm intelligence; IoT**

I. Introduction

"Swarm computing", an approach inspired by the collaborative and decentralized behavior of some systems in nature [1], gains particular importance in the context of the Internet of Things (IoT). With around 80 billion connected devices in the world by 2025 [2], the collective "intelligent" behavior will synergistically emerge from the "swarm" of devices and their collaboration. In this context, we consider an exciting new computing paradigm called *Adaptive Swarm Intelligence* (ASI). It refers to an envisioned future where computing devices (or "agents") work together collaboratively (in swarm mode) to provide intelligent services and improve their learning capabilities *in the field*. A specific example domain is that of connected/autonomous cars. Other application domains include network cybersecurity, where the detection of denial-of-service attacks is done via localized node-to-node information exchange; unmanned aerial vehicles, for example for search-and-rescue operations; and resource management (like power consumption) in many-core chips and datacenters.

ASI allows devices to exchange "knowledge" and reach consensus in real time for a particular situation or task to solve. This crowdsourced decision constitutes a form of automatic data labeling that allows devices to keep updating their on-board AI/ML models in an unsupervised manner. Leveraging the collaborative swarm mode also reduces real-time deadline pressures at the individual agent level, while improving overall resilience through redundancy. As a result, ASI emerges as a key enabler for cognitive systems that do not break in the field, but that keep learning continuously from in-field data. We believe that IoT solutions that do not consider swarm operation will fall short of the level of adaptation required to operate properly in highly-dynamic environments.

In this paper, we present a specific ASI case study for cooperative sensor fusion in prospective connected/autonomous vehicles, which constitutes the driving application of the IBM-led "Efficient Programmability of Cognitive Heterogeneous Systems" (EPOCHS) project under the DARPA DSSoC program. We focus on the *EPOCHS Reference Application* (ERA) for multi-vehicle sensor fusion and show characterization results on a x86 system that allow us to draft preliminary conclusions about ERA's performance characteristics and real-time needs.

II. The Need for Adaptive Swarm Intelligence

Adaptive Swarm Intelligence (ASI) is expected to find a breeding ground in many domains, including network cybersecurity, connected/autonomous vehicles, robotics and cyber-physical systems in general. In the specific context of connected/autonomous cars, the benefits are large: for example, one of the most critical challenges that the automotive industry faces these days is related to *false predictions* while the car is driving and perceiving the environment. False predictions (either false negatives or false positives) can be alleviated through the use of arrays of sensors to build redundancy into the self driving system. However, this approach can be economically inefficient and not necessarily fix the problem. In some cases, increasing the on-board sensing and computation capabilities does not necessarily reduce the false prediction rate by the same factor. ASI, on the other hand, can effectively overcome this problem, since redundancy is inherent to the vehicle swarm. Although individual cars may have a relatively poor vision of the surrounding environment (for example,

DISTRIBUTION STATEMENT A. Approved for public release: distribution is unlimited. This research was developed, in part, with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

due to bad weather conditions or line-of-sight occlusion), the fusion of their locally-generated "views" in real time can result in more reliable navigation due to the reduction or virtual elimination of false predictions as Figure 1 sketches. The rationale behind this observation comes from the *swarm robotics* domain. Previous works have shown that the *success* to complete a *task* by a group of robots improves with the number of robots in the swarm. For example, object recognition accuracy increases significantly when performed in a cooperative manner and with a relatively large number of robots involved in the process [3]. Similarly, navigation delays are cut down when the navigation activities are conducted cooperatively by a robot swarm [4]. We anticipate the emergence of a similar behavior in groups of vehicles when the proper swarming elements are in place.

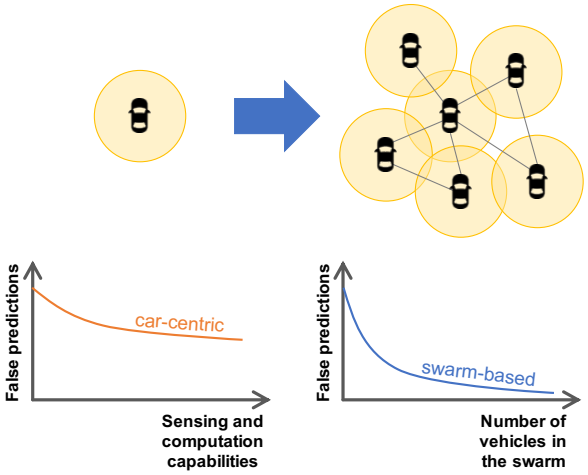

Fig. 1: Increasing the on-board sensing and computation capabilities does not necessarily reduce false predictions by the same factor. ASI is expected to overcome this problem.

But why ASI *now*? Beginning in the 90s, chip processors kept steadily shrinking in size and form factor while growing in computation capabilities, driven by Moore's law and very-large-scale integration. This phenomenon constituted a key enabler that sparked the mobile computing revolution. Later on, during the 2000s, sensor technology became cheap and pervasive. Small-scale processors alongside pervasive sensing gave rise to the IoT *boom*. Today, wireless communication technology (including 5G and *dedicated short-range communication* protocols) is expected to reach performance levels that will enable ultra-fast, ultra-high-bandwidth, low-latency wireless device-to-device connectivity. We believe that these three key technical enablers (*small-scale processors*, *pervasive sensing* and *ultra-fast connectivity*) will give rise to the ASI revolution (Figure 2).

III. Cooperative Sensor Fusion: A Case Study

The self-driving technology deployed in autonomous vehicles is still in a nascent stage and far from being reliable enough for use in real scenarios. The data feeds generated by the on-board sensing system (including radars, lidars, or cameras) can be erroneous in some cases, resulting in

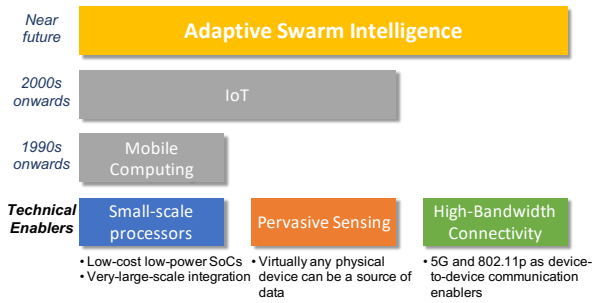

Fig. 2: Small-scale processors, pervasive sensing and high-bandwidth connectivity are the key enablers for the next big revolution: *Adaptive Swarm Intelligence*.

false predictions that can jeopardize the overall safety and confidence in autonomous navigation. Recent infamous examples of this include a Tesla car that struck a tractor-trailer in Florida in May 2016, probably due to lighting or other imaging issues that prevented the computer from detecting the obstruction ahead [5]. More recently, in March 2018, the self-driving system software in an Uber vehicle failed to identify a pedestrian [6]. Sadly, these are only two examples among several cases that indicate that the reliability of the self-driving software/hardware platform still requires groundbreaking approaches to reduce or eliminate false predictions.

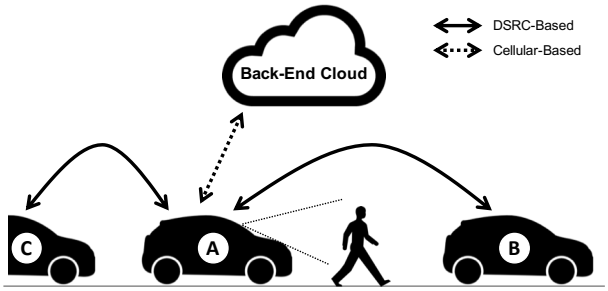

Fig. 3: Illustrative autonomous driving scenario where automobiles depend on reliable and fast V2V communications.

Usually, automakers use arrays of sensors to build redundancy and resolve ambiguity in self driving cars. However, increasing the number of sensors not necessarily reduces the false prediction rate (imagine a vehicle which line-of-sight is obstructed by another vehicle or object) and can also result in cost-prohibitive solutions. ASI, on the other hand, constitutes a complementary level of redundancy that can effectively improve the robustness of the self-driving system via multi-vehicle (cooperative) sensor fusion, as the authors have preliminarily discussed in [7]. Specifically, we envision a solution where cars build and exchange locally-generated surrounding maps. Each vehicle then "fuses" its local map with the ones received from other nearby vehicles in real time. The surrounding map resulting from the fusion process is expected to be more reliable due to the reduction or virtual elimination of false predictions. In this vehicle-to-vehicle (V2V) information exchange process, communications

play a critical role due to the highly-dynamic nature of the real-time scenarios of interest. Figure 3 depicts an illustrative case study where connected vehicles interact to each other through dedicated short-range communication (DSRC) [8] to enable *cooperative perception*. The ultimate goal for a given car is to generate reliable views of its surroundings in real time and with the support of other nearby vehicles.

In this context, our group has been recently awarded with a research grant from DARPA to develop a hardware-software platform for real-time swarm-based sensor fusion in autonomous vehicles. Our project called "Efficient Programmability of Cognitive Heterogeneous Systems" (EPOCHS), involves multi-modal sensing, local map generation, map exchange through vehicle-to-vehicle communication, and real-time multi-vehicle map fusion (Figure 4). The EPOCHS platform will also include a variety of novel features, like mechanisms for power efficiency and reliability, a "smart" operating system scheduler, latest generation compilation toolchains, and methodologies for rapid chip prototyping and development. In the next sections, we focus on one specific piece of the EPOCHS project: the *EPOCHS Reference Application* (ERA) for multi-vehicle sensor fusion.

IV. ERA: EPOCHS Reference Application

The EPOCHS Reference Application (ERA) will serve as the testbed and driving application domain for the development of our EPOCHS methodology and associated technologies. The chosen ERA domain implements a software platform to enable multi-vehicle (cooperative) sensor fusion in future autonomous/connected cars, using elements of computer vision, navigation, and V2V communications. In this context, multi-vehicle sensor fusion constitutes a concrete use case of the broader ASI paradigm.

ERA is composed of a *Sensing and Mapping Fabric* —which implements on-board multi-modal sensing, sensor data processing, and mapping— and a *Communication and Consensus Fabric* —in charge of V2V communications and multi-vehicle map fusion (Figure 4). Note that the results of the *Real-Time Consensus & Fusion* stage are fed to other action engine and vehicle control blocks which are not part of ERA. The *Sensing and Mapping Fabric* builds upon the Robot Operating System (ROS) [9] to handle the environment sensing and map creation tasks. The *Communication and Consensus Fabric* uses the GNU Radio framework [10] to implement the MAC and PHY layers of the IEEE 802.11p standard [8] for dedicated short-range communications (DSRC) between vehicles. Finally, we plan to develop custom software to implement the inter-vehicle map fusion task. The current version of ERA includes a subset of the functionalities depicted in Figure 4, enough to compose an end-to-end working ERA version that can be used by other task groups within our EPOCHS project.

A. Current ERA Version

The current ERA version 1, which was released in October 2018 as open-source code [11], implements a subset of the functionalities presented in Figure 4 – specifically:

- Video data capturing using an on-board depth camera model.

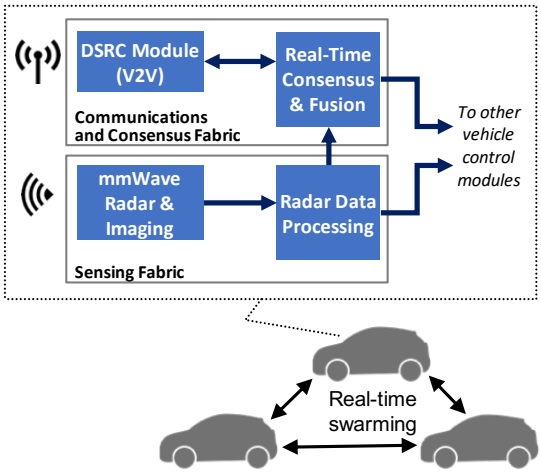

Fig. 4: Swarm-based sensor fusion in connected/autonomous vehicles as part of our DARPA-sponsored EPOCHS project.

- Real-time generation of 2D occupancy maps using the captured video data.
- Partial GNU Radio-based implementation of the IEEE 802.11p standard leveraging open-source software [12], [13].

For convenience during this initial phase, we adopted Gazebo (a 3D virtual environment for simulating complete robot applications with detailed physics) to model the vehicular environment and its physical aspects [14]. Specifically, we mimic vehicles using the *TurtleBot* robot model provided by Gazebo. This robot model can be easily configured with a variety of built-in sensor models; in our particular case, we equipped each TurtleBot with an on-board depth camera.

Figure 5 presents a block diagram of the currently implemented functionalities and their interconnections from a single vehicle's (or robot's) perspective. As described above, Gazebo simulates the physical robotics "world" including the sensing capabilities. The raw data provided by the modeled depth camera is forwarded to ROS for real-time generation of 2D occupancy maps using the *costmap_2d* package. In this context, a 2D occupancy map is a grid of cells centered in the robot's current location where each cell has an associated state value: *free*, *occupied*, or *unknown*. Occupancy maps are compressed and serialized, and encoded for DSRC/802.11p transmission. ERA version 1 does not yet support over-the-air communication between robots (or vehicles). Instead, the Orthogonal Frequency Division Multiplexing (OFDM) symbols generated by the DSRC/802.11p transmitter are sent through a UDP socket to a "dummy" remote agent (a simple C program) that bounces the symbols back to the robot. The received OFDM symbols are decoded by the DSRC/802.11p receiver, decompressed and unserialized, and finally compared against the originally sent map for verification purposes.

B. Performance Characterization

ERA is a relatively complex piece of software due to the coexistence (and associated engineering challenges) of Gazebo, ROS and GNU Radio in the same platform, and

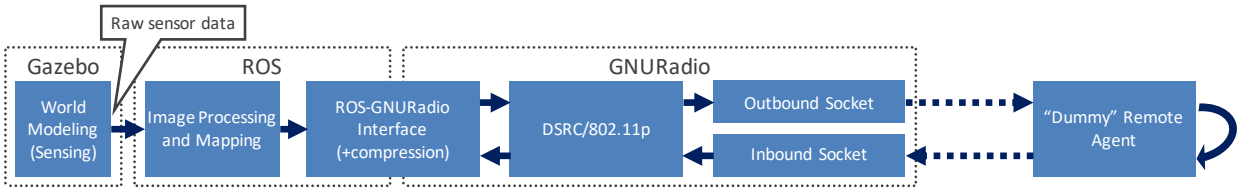

Fig. 5: ERA version 1 block diagram.

the programming language heterogeneity across the different components (with blocks written in C, C++ and Python). Therefore, it becomes imperative to define and execute a thorough ERA performance characterization campaign to (1) debug and ensure its proper operation, and (2) to study its *hotspots* and discover potentially accelerable code portions. At present we conduct simultaneous efforts to port and characterize ERA on x86, ARM and POWER architectures. This process will also include the porting of specific ERA components (identified hotspots) to GPU and/or FPGA for acceleration, as part of the future work.

In this section, we focus on the GNU Radio components within ERA (leaving ROS aside for the moment) to identify the most critical (CPU intensive) blocks in the 802.11p transceiver (Figure 6) running on a x86 architecture. We use the Linux *perf* [15] performance analysis tool, with plans to make use of GNU Radio's built-in performance counters [16] in the near future. We define two GNU Radio running settings: single instruction, multiple data (SIMD) vectorization *disabled* and *enabled*. GNU Radio supports SIMD vectorization through VOLK (Vector-Optimized Library of Kernels), a library of hand-written SIMD code for different mathematical operations, introduced as a part of GNU Radio in 2010 [17]. Table I summarizes the most important elements of our evaluation platform.

TABLE I: x86 evaluation platform.

CPU	Quad-core Intel Core i7 "Ivy Bridge" (22nm)
OS	Ubuntu 16.04
SDR Framework	GNU Radio 3.7.9, Volk 1.2.3
802.11p Parameters	BPSK 1/2 (encoding), 10 MHz (bandwidth), 5.89 GHz (frequency)

Figure 7 shows the execution time fraction when the GNU Radio/802.11p transceiver runs on a x86 platform *without* and *with* SIMD vectorization ("Volk Disabled" and "Volk Enabled", respectively). We observe that (1) the *mapper* block uses the CPU significantly more time than the rest of the blocks, and (2) that the execution profile of this particular 802.11p implementation is not affected by SIMD vectorization. The *mapper* block is not optimized to make use of SIMD support; and given that this block governs ERA's execution time, vectorization does not significantly affect overall performance. *Mapper* is only part of the 802.11p transmitter and, therefore, it is not one of the most critical components to accelerate (the transmitter does not pose high computational demands, as the whole frame can be pre-computed before it is streamed to the radio [13]). This block is responsible for the generation of the MAC header and the 32-bit Cyclic Redundancy Check (CRC) for error detection; it also takes care of convolutional encoding, puncturing, mapping to one of the supported complex constellations (BPSK, QPSK, 16-QAM, or 64-QAM), and interleaving of pilot symbols. We plan to dive deeper into the implementation of this block to assess acceleration alternatives, including vectorization (to leverage Volk) or GPU/FPGA acceleration (if available).

Another relevant metric is the speed at which the receiver can process incoming samples, which is an indication of the receiver's real-time processing capabilities. We executed a simple experiment varying the rate at which samples are injected into the receiver, in order to determine the saturation point — i.e. the moment at which the receiver cannot keep up with the incoming sample stream. Figure 8 presents the number of 1500-byte samples that the receiver can process (decode) per second as a function of the number of 1500-byte samples that are injected per second, when the transceiver runs on a x86 platform *without* and *with* SIMD vectorization. At first glance, we observe that the receiver's computational performance scales linearly with the injection rate up to 50 Hz, and it is still good enough at 100 Hz. This is probably acceptable for any realistic real-time application in the context of connected/autonomous vehicles.

Figures 7 and 8 also confirm that the use of Volk does not make any difference for this particular transceiver implementation on the x86 architecture used for the evaluation. We dove deeper into the transceiver code using Linux *perf* to identify where Volk is leveraged and how frequently. As expected, only a few GNU Radio blocks make use of it (complex_to_mag, conjugate_cc, fft_vcc_fftw, multiply_cc and sync_long) for relatively short periods of time, as Table II shows.

TABLE II: GNU Radio blocks that make use of Volk kernels, and their corresponding execution time fractions (global).

Block	Volk Kernel	Time Fraction
complex_to_mag	volk_32fc_magnitude_32f_u_sse	0.00%
	volk_32fc_magnitude_squared_32f_u_avx	0.61%
conjugate_cc	volk_32fc_conjugate_32fc_a_sse3	0.00%
	volk_32fc_conjugate_32fc_u_avx	0.00%
fft_vcc_fftw	volk_32fc_32f_multiply_32fc_generic	1.67%
multiply_cc	volk_32fc_x2_multiply_32fc_a_avx	0.78%
sync_long	volk_32fc_x2_dot_prod_32fc_a_avx	0.00%

The transceiver presented in Figure 6 shows the existence of two FFT blocks: a *forward* FFT in the receiver and a *reverse* (inverse) one in the transmitter. They are both the same size (64 elements). As part of the characterization process, we plan

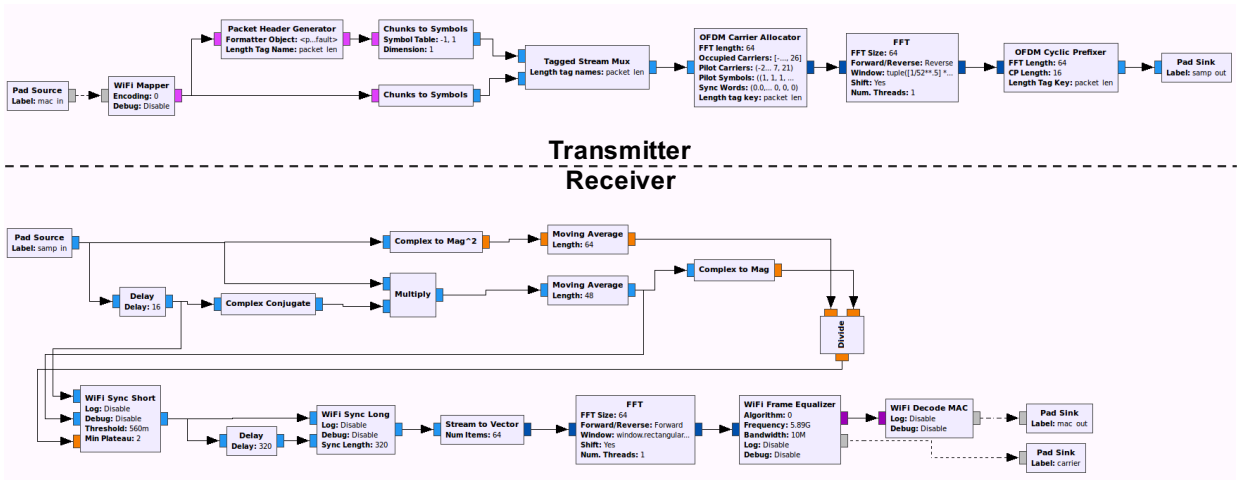

Fig. 6: GNU Radio-based 802.11p transceiver flowgraph [12], [13].

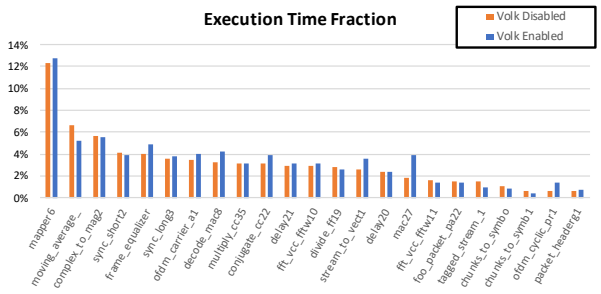

Fig. 7: Execution time fraction of GNU Radio/802.11p on x86.

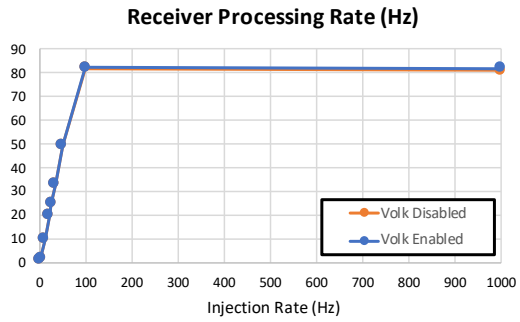

Fig. 8: Processing rate of GNU Radio/802.11p receiver on x86.

to analyze the FFT blocks to determine (1) their criticality in the performance of the 802.11p transceiver, and (2) if a single time-multiplexed hardware FFT accelerator would suffice — and if so, devise the most effective FFT scheduling mechanism to make use of the single accelerator.

The performance characteristics of the 802.11p transceiver presented so far suggest that the adopted x86 platform is good enough to meet the real-time requirements for vehicle-to-vehicle (V2V) communications. Given that this is not a system intended for embedded applications, we are conducting equivalent measurements on a low-power quad-core ARM Cortex-A53 processor, which results will be presented in future publications.

C. Roadmap and Future Work

Figure 9 presents a tentative timeline for the development of ERA and the introduction of new capabilities. ERA version 1 (released in October 2018) runs on a single computer with simulated scenarios and sensor data, and 2D occupancy maps exchanged over UDP sockets. We are currently working on ERA version 2, in which the simulated robots will be decoupled across multiple (e.g. two) computer systems and maps will be exchanged through over-the-air communication using appropriate Universal Software Radio Peripheral (USRP) devices [18]. In version 2, we still simulate vehicles using TurtleBot models in Gazebo. After the release of version 2 (and before the completion of version 3), we plan to augment the robots' sensing capabilities using convolutional neural networks (CNNs) to detect and classify surrounding objects. The goal is to include information about the detected objects in the occupancy maps cells, in addition to the currently used *free*, *occupied* and *unknown* state values. In other words, occupied cells will be populated with the corresponding objects' identities. Finally, we expect to transition to real mobile devices (e.g. real TurtleBots or small "toy" cars) in ERA version 3, which will preserve all the functionalities developed in the two previous versions. As a result, version 3 will get rid of Gazebo and, in its place, it will make use of real sensors (e.g. cameras) for data capturing.

Another important aspect that we are looking into is *security*. The current ERA version 1 is built on ROS "1", which has a custom-built communication protocol that is not designed for multi-agent or safety critical systems. As a result, we are assessing the partial or total migration from ROS "1" to ROS "2", the next generation ROS built with modern requirements in mind. For example, the communication layer adopts Data Distribution Service (DDS) [19], a standardized protocol used in automation, military, and aerospace [20]. DDS provides DDS-Secure, a safety extension bringing authentication, access control, and encryption to the protocol. We are considering a gradual transition to ROS "2". Initially, both versions can

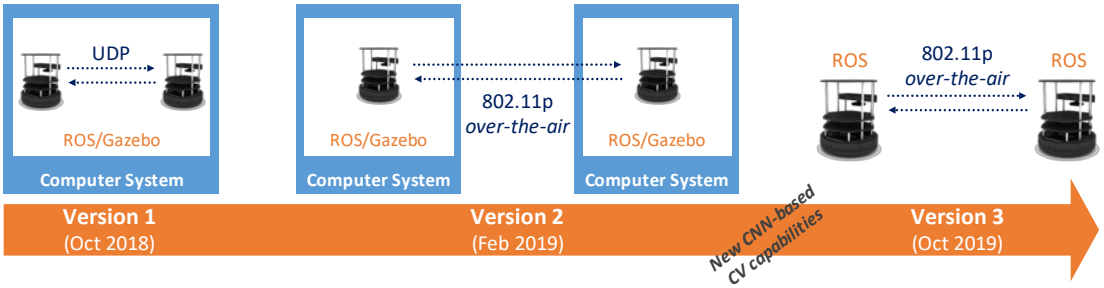

Fig. 9: Tentative ERA development timeline.

exist and run side by side and communicate with each other through *ros1_bridge* (each environment's topics and services are visible to each other through this bridge). ERA specific code will be ported to ROS "2" (as much as the data structures and the underlying libraries permit), while simulation code will remain in ROS "1".

Conclusions

We present a bio-inspired paradigm called *Adaptive Swarm Intelligence* (ASI) where heterogeneous edge devices engage in collaborative "swarm" computing for robust and adaptive real-time operation. ASI allows swarm agents to work together collaboratively to improve their learning capabilities, with applications in connected/autonomous cars, network cyber-security, unmanned aerial vehicles, and distributed resource management in many-core chips and datacenters. As a result, ASI emerges as a key enabler for cognitive systems that do not break in the field, but that keep learning continuously from in-field data.

In this context, communications play a critical role due to the highly-dynamic nature of the real-time scenarios of interest. This paper presents a specific ASI case study for co-operative sensor fusion in prospective connected/autonomous vehicles, which constitutes the driving application of the IBM-led "Efficient Programmability of Cognitive Heterogeneous Systems" (EPOCHS) project under the DARPA DSSoC program. We focus on the *EPOCHS Reference Application* (ERA) for multi-vehicle sensor fusion and show characterization results on a x86 system that allow us to draft preliminary conclusions about ERA's performance characteristics and real-time needs.

Acknowledgment

We are grateful to all our colleagues within the IBM-led project titled EPOCHS: Efficient Programmability of Cognitive Heterogeneous Systems, sponsored by DARPA under its DSSoC program. This includes researchers from Columbia University, Harvard University and University of Illinois at Urbana-Champaign — in addition, of course, to many other IBM research personnel, including Dr. Jeff Burns, who has been an inspirational supporter of the ASI project. A special acknowledgment to Dr. Akin Sisbot for his active participation in the development of the *EPOCHS Reference Application* (ERA).

References

[1] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY, USA: Oxford University Press, Inc., 1999.

[2] M. Kanellos, "152,000 smart devices every minute in 2025: IDC outlines the future of smart things," http://www.forbes.com/sites/michaelkanellos/2016/03/03/152000-smart-devices-every-minute-in-2025-idc-outlines-the-future-of-smart-things/#1d4cd6b069a7, March 2016.

[3] A. Giusti, J. Nagi, L. Gambardella, and G. D. Caro, "Cooperative sensing and recognition by a swarm of mobile robots," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 551–558.

[4] F. Ducatelle, G. A. Di Caro, A. Förster, M. Bonani, M. Dorigo, S. Magnenat, F. Mondada, R. O'Grady, C. Pinciroli, P. Rétornaz, V. Trianni, and L. M. Gambardella, "Cooperative navigation in robotic swarms," *Swarm Intelligence*, vol. 8, no. 1, pp. 1–33, Mar 2014.

[5] The New York Times, "Self-driving Tesla was involved in fatal crash, U.S. says," https://www.nytimes.com/2016/07/01/business/self-driving-tesla-fatal-crash-investigation.html, 2016.

[6] National Transportation Safety Board (NTSB), "Preliminary report released for crash involving pedestrian, Uber Technologies, Inc., test vehicle," https://www.ntsb.gov/news/press-releases/Pages/NR20180524.aspx, 2018.

[7] A. Vega, A. Buyuktosunoglu, and P. Bose, "Towards "smarter" vehicles through cloud-backed swarm cognition," in *Proceedings of the 29th IEEE Intelligent Vehicles Symposium*, ser. IV '18, 2018.

[8] Wikipedia, "IEEE 802.11p — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=IEEE_802.11p&oldid=774355529, 2017.

[9] Open Source Robotics Foundation, "Robot Operating System (ROS)," http://www.ros.org, 2018.

[10] GNU Radio Foundation, Inc., "GNU Radio," https://www.gnuradio.org, 2018.

[11] IBM Corp., "EPOCHS Reference Application (ERA)," https://github.com/IBM/era, 2018.

[12] B. Bloessl, "IEEE 802.11 a/g/p transceiver for GNU Radio," https://github.com/bastibl/gr-ieee802-11, 2018.

[13] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "Performance Assessment of IEEE 802.11p with an Open Source SDR-based Prototype," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1162–1175, May 2018.

[14] Open Source Robotics Foundation, "Gazebo," http://gazebosim.org, 2018.

[15] Linux Kernel Organization, "perf: Linux profiling with performance counters," https://perf.wiki.kernel.org, 2018.

[16] GNU Radio Foundation, Inc., "GNU Radio Performance Counters," https://wiki.gnuradio.org/index.php/PerformanceCounters, 2018.

[17] ——, "Vector-Optimized Library of Kernels," http://libvolk.org, 2018.

[18] Ettus Research, "Ettus USRP B200," https://www.ettus.com/product/details/UB200-KIT, 2018.

[19] Open Source Robotics Foundation, "ROS on DDS," https://design.ros2.org/articles/ros_on_dds.html, 2018.

[20] The Object Management Group, "Who's Using DDS?" https://www.omgwiki.org/dds/who-is-using-dds-2/, 2018.