Chapter 38
Incremental Distributed Learning With JavaScript Agents for Earthquake and Disaster Monitoring

Stefan Bosse
University of Bremen, Germany

ABSTRACT

Ubiquitous computing and The Internet-of-Things (IoT) grow rapidly in today's life and evolving to Self-organizing systems (SoS). A unified and scalable information processing and communication methodology is required. In this work, mobile agents are used to merge the IoT with Mobile and Cloud environments seamless. A portable and scalable Agent Processing Platform (APP) provides an enabling technology that is central for the deployment of Multi-Agent Systems (MAS) in strong heterogeneous networks including the Internet. A large-scale use-case deploying Multi-agent systems in a distributed heterogeneous seismic sensor and geodetic network is used to demonstrate the suitability of the MAS and platform approach. The MAS is used for earthquake monitoring based on a new incremental distributed learning algorithm applied to seismic station data, which can be extended by ubiquitous sensing devices like smart phones. Different (mobile) agents perform sensor sensing, aggregation, local learning and prediction, global voting and decision making, and the application.

INTRODUCTION

The IoT and mobile networks emerge in today's life and are becoming part of pervasive and ubiquitous computing networks with distributed and transparent services. One major goal is the integration of sensor networks in the Internet and Cloud environments, with emerging robustness and scalability requirements. Robustness and scalability can be achieved by self-organizing and self-adaptive systems (self-*). Agents are already deployed successfully in sensing, production, and manufacturing processes, proposed by Caridi (2000), and newer trends poses the suitability of distributed agent-based systems for the control

DOI: 10.4018/978-1-5225-6195-8.ch038

of manufacturing processes, shown by Pechoucek (2008), facing manufacturing, maintenance, evolvable assembly systems, quality control, and energy management aspects, finally introducing the paradigm of industrial agents meeting the requirements of modern industrial applications by integrating sensor networks. Mobile Multi-agent systems can fulfill the self-organizing and adaptive (self-*) paradigm, shown in Bosse (2015A). Distributed data mining and Map-Reduce algorithms are well suited for self-organizing MAS. Cloud-based computing with MAS means the virtualization of resources, i.e., storage, processing platforms, sensing data or generic information, discussed by Lehmhus (2015). Mobile Agents reflect a mobile service architecture. Commonly, distributed perceptive systems are composed of sensing, aggregation, and application layers, shown in Figure 1, merging mobile and embedded devices with the Cloud paradigm as in Lecce (2013). But generic Internet, IoT, and Cloud environments differ significantly in terms of resources: The IoT consists of a large number of low-resource or mobile devices interacting with the real world. The devices have strictly limited storage capacities and computing power, and the Cloud consists of large-scale computers with arbitrary and extensible computing power and storage capacities in a basically virtual world. A unified and common data processing and communication methodology is required to merge the IoT with Cloud environments seamless, which can be fulfilled by the mobile agent-based computing paradigm, discussed in this work.

The scalability of complex ubiquitous applications using such large-scale cloud-based and wide area distributed networks deals with systems deploying thousands up to million agents. But the majority of current laboratory prototypes of MAS deal with less than 1000 agents, posed by Pechoucek (2008). Currently, many traditional processing platforms cannot yet handle a big number of agents with the robustness and efficiency required by the industry (Pechoucek (2008)), sensing, and Cloud applications. In the past decade, the capabilities and the scalability of agent-based systems have increased substantially, especially addressing efficient processing of mobile agents. The integration of perceptive and mobile devices in the Internet raises communication and operational barriers, which must be overcome by a unified agent processing architecture and framework. In this work, the JavaScript Agent Machine (JAM) is used, supporting mobile JavaScript agents (AgentJS). JAM is entirely written in JavaScript, which can be executed on a wide variety of host platforms including WEB browsers. The platform is discussed in detail in Bosse (2016B). Lecce (2013) concluded that a sensor network as part of the IoT is composed of low-resource nodes. Smart systems are composed of more complex networks (and networks of networks) differing significantly in computational power and available resources, raising inter-communication barriers. These smart systems unite sensing, aggregation, and application layers, Lecce (2013), shown in Figure 1, requiring a unified design and architecture approach. Smart systems glue software and hardware components to an extended operational unit, the basic cell of the IoT. Mobile AgentJS agents can operate in such strongly heterogeneous environments by using JAM and a new low-resource JavaScript engine JVM, suitable for embedded systems, in contrast to common approaches using Java-based frameworks (e.g., JADE and Jason, Bellifemine (2007)) or object-oriented systems like SWARM by Minar (1996).

Machine Learning (ML) is used in a wide range of fields for state prediction or recognition. The concept of ML is closely related to the agent behaviour model, basically consisting of perception, planning (reasoning), and action. In recent years ML gets attraction for earthquake prediction and monitoring, supporting disaster management, e.g., using neuronal networks shown by Alarifi (2012).

The JAM platform provides ML as a service for agents. Commonly, ML used to derive a classification model is performed centralized, i.e., all input data is collected and processed by one learner instance. For large scale distributed systems such a central instance is not suitable, is not conforming to the MAS architecture, and introduces a single point of failure. In Bosse (2016C), a first attempt was made to