Safe, Secure Executions at the Network Edge : Coordinating Cloud, Edge, and Fog Computing

Mäkitalo, Niko

2018-01

Mäkitalo , N , Ometov , A , Kannisto , J , Andreev , S , Koucheryavy , Y & Mikkonen , T 2018 , ' Safe, Secure Executions at the Network Edge : Coordinating Cloud, Edge, and Fog Computing ' , IEEE Software , vol. 35 , no. 1 , 0740-7459 , pp. 30-37 . https://doi.org/10.1109/MS.2017.4541037

http://hdl.handle.net/10138/231350
https://doi.org/10.1109/MS.2017.4541037

acceptedVersion

@ARTICLE{makitalo2018executions,
author={N. Mäkitalo and A. Ometov and J. Kannisto and S. Andreev and Y. Koucheryavy and T. Mikkonen},
journal={IEEE Software},
title={Safe, Secure Executions at the Network Edge: Coordinating Cloud, Edge, and Fog Computing},
year={2018},
volume={35},
number={1},
pages={30-37},
keywords={Computational modeling;Cyber-physical systems;Programming;Safety;Security;Servers;Software;edge computing;fog computing;programmable world;programming models;proximate connectivity;security and trust;software development;software engineering;systems and software design},
doi={10.1109/MS.2017.4541037},
ISSN={0740-7459},
month={January},
}

Safe and Secure Execution at the Network Edge: A Framework for Coordinating Cloud, Fog, and Edge

Niko Mäkitalo, Aleksandr Ometov, Joona Kannisto,
Sergey Andreev, Yevgeni Koucheryavy, and Tommi Mikkonen

Abstract

System design where cyber-physical applications are securely coordinated from the cloud may simplify the development process. However, all private data are then pushed to these remote 'swamps', and human users lose the actual control as compared to when the applications are executed directly on their devices. At the same time, computing at the network edge is still lacking support for such straightforward multi-device development, which is essential for a wide range of dynamic cyber-physical services. In this work, we propose a novel programming model as well as contribute the associated secure connectivity framework for leveraging safe coordinated device proximity as an additional degree of freedom between the *remote cloud* and the *safety-critical network edge,* especially under uncertain environment constraints.

Keywords: Edge computing, programmable world, programming models, proximate connectivity, systems and software design.

1 Motivation and Background

Today, the cloud has evolved into a ubiquitous solution for enterprises in their quest for a unified digital platform. To this end, it represents a centralized infrastructure that has become the control point to manage computing power, storage, processing, integration, and decision-making assets for modern corporations. Boosting these processing and decision making capabilities even further, there is a need to offer novel high-confidence technologies that will have the capability to support billions of networked devices, as we are stepping into the era of interconnected Cyber-Physical Systems (CPSs) [1].

The particularly challenging operating conditions of future CPSs are represented by the areas of poor or unavailable infrastructure network connectivity, which need to be handled to maintain sustainability and enable faster decision-making based on localized data intelligence across heterogeneous system components. These considerations are underpinning the recent trend to transition from the centralized cloud platforms toward the network edge, which is essentially dispersing the cloud back to the origin – the end devices [2]. With advanced routers, gateways, microservices, containers, and APIs, it is increasingly feasible to execute these smaller, self-contained, and purpose-driven services that specifically target certain dedicated functions required near the edge.

The distinct computing paradigms, such as 'cloud' (computations on a remote server), 'fog' (computations at the local area network level), and 'edge' (computations on end-devices) computing, act similarly under conventional use cases, but become fundamentally different when considering safety-critical CPSs that operate in dynamic and uncertain conditions [3]. Examples include overtaking control of a moving vehicle and hacking an industrial robot, where various vulnerabilities may be exploited to hijack remote access to the capable factory equipment.

To mitigate these vulnerabilities, new systems and software engineering methods are required where networked machines can interact more freely, by forming connections according to the actual demand and not because this is requested by the central managing entity. This calls for revisiting the ways of interaction between the devices and their operating environment, which becomes the target of this work. Its main contributions thus are an Action-Oriented Programming model and the associated framework that can dynamically adapt to the edge and the cloud according to particular environment and connectivity conditions. Further, it is compared to mobile app-based and cloud-based CPS deployments. Finally, a framework to enable secure coalitions and dynamic management of collective executions is also outlined.

2 Sensing and Actuation Executions at the Network Edge

Billions of smart CPS devices at the network edge require proximity-based communication *mostly* with the cloud connectivity, but these two aspects have traditionally been addressed in isolation [4]. At the same time, the lion's share of CPS interactions is still about their users (e.g., the concept of quantified self [5]), and the human element is tightly involved into the decision-making process. The capability to switch from the cloud to the edge (or the app) dynamically, based on the operating conditions and user requests, allows to control device behavior more efficiently. In what follows, we focus on advancing software development efforts, so that the said activities are executed on the network edge to boost the dynamic adaptation of a CPS to complex environment constraints, while remaining mindful of human perception.

2.1 Action-Oriented Programming Model

The proposed Action-Oriented Programming (AcOP) model builds upon the following constructs. It roots in the so-called Social Devices paradigm [6], and the original concept is tailored here to the context of edge computing as driven by the CPS evolution. The new model is realized on top of JavaScript programming language, which enables deployment on a vast number of hardware platforms.

Sensation – An input coming from the physical, cyber, or social world.

Instrumental to CPSs is observation of various events coming from the outer world, and then acting upon these events. In AcOP, these observed events are named *sensations*. The abstraction level of the sensations may vary, and in addition to observing physical world's phenomena, the processes in cyber and social worlds can be monitored as well. A concrete example of a sensation is the changed sensor value, while a more abstract sensation is, e.g., when a friend is nearby, which combines data from different worlds (Facebook friendship and Bluetooth signal strength values).

Capability – Physical objects as programmable JavaScript objects.

In AcOP, physical objects and digital (micro)services constitute programmable JavaScript objects. They are described with AcOP *capabilities* that define the ways in which a certain machine can interact with other machines and humans. An example here is the *talking* capability where the device is able to translate text into speech. The capabilities produce abstract *capability sensations*, such as "temperature has changed" or "coffee is ready", which are derived from raw sensations. These help the developers define scheduling policies for collective executions.

Action – Joint behavior of machines and humans.

In the heart of our model are *actions* that define joint operations across multiple devices and people. Typically, an action is a modular unit that determines how several devices interact with each other over a certain period of time. Such actions are defined with JavaScript and comprise three parts, a *casting method* used for selecting participating devices based on their capabilities, an *enabling condition* and a *body,* used for programming the interactions by utilizing the AcOP capabilities as well as the basic programming logic. The modularity of the actions helps make them more generic, so that they may be exploited in many different executions; similarly, the device capabilities may be employed by many actions.

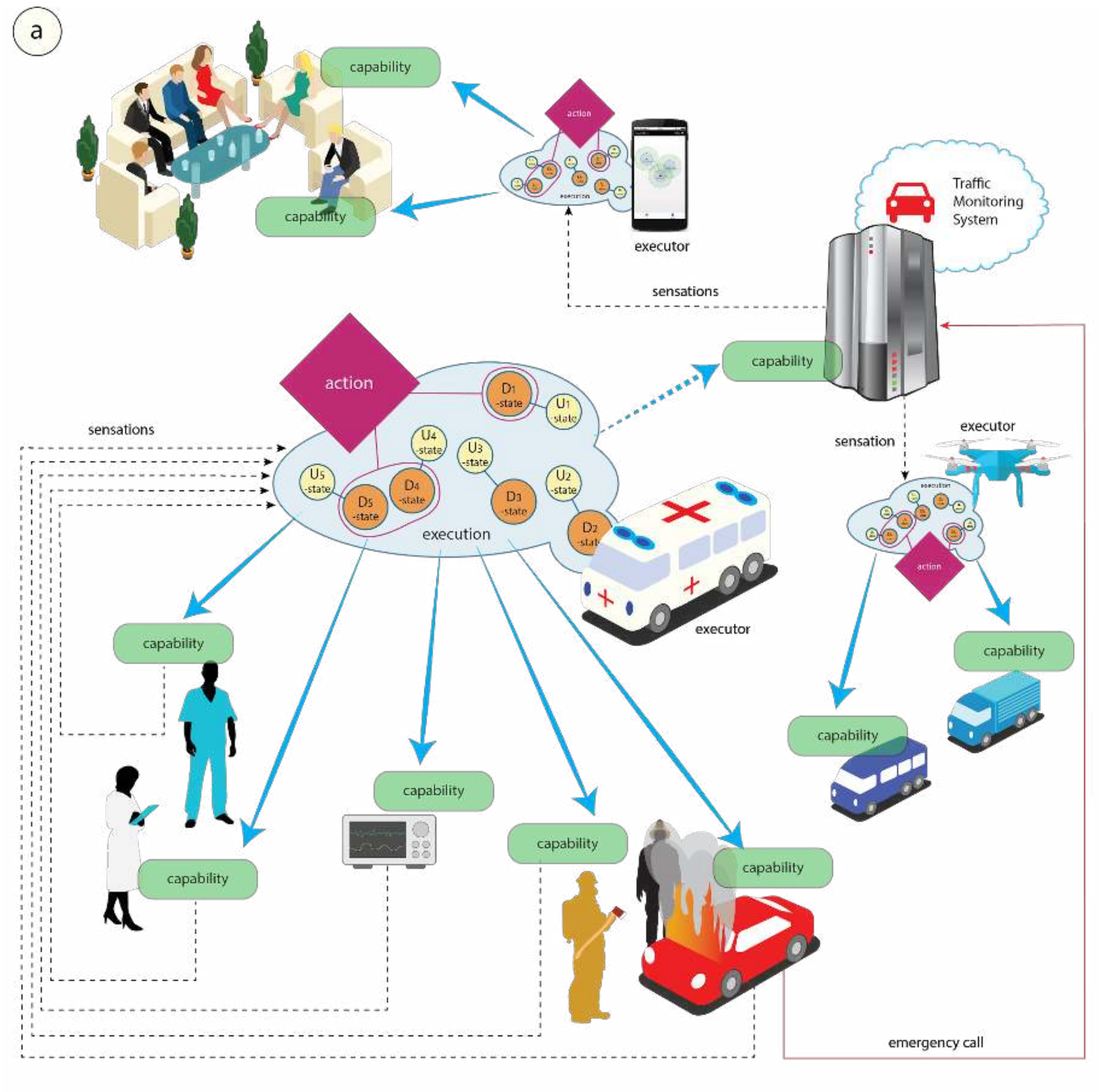
Collective Execution – Coalition of trusted entities sensing and acting toward a common goal.

In AcOP, a set of machines and humans form *coalitions* by engaging into trust negotiations. These coalitions then collectively execute software where machines and humans interact and cooperate. The key idea of the collective execution is detecting and maintaining information about the sensations coming from the various worlds as well as from the coalition participants. Then, collective execution attempts to *schedule an action* for a set of devices, which are selected for specific roles in this action based on their capabilities and properties. In practice, collective execution operates so that one device in a coalition at a time assumes the role of coordinator and then executes the code responsible for scheduling the actions. At the same time in the background, all of the other devices in a coalition contribute by exchanging information that is essential for that particular execution via secure connections.

2.2 Traffic Emergency Context: AcOP Operation at the Network Edge

As a characteristic use case, we study the scenario where contemporary vehicles can call 911 in emergency situations. Consider Fig. 1.a depicting a traffic accident, in which the car involved would immediately report to the traffic monitoring system. The vehicle or the traffic system also communicates the supporting information – or a *sensation* – to a set of *executions*. For instance, *Fire Department's execution* may detect the accident and schedule an action to leverage the car's *talking capability* to poll passengers whether they are unharmed as well as provide instructions. The execution can also schedule further actions, e.g., command the traffic monitoring system to send a drone to analyze if a fire has started, call an ambulance if an injury was suspected [7], etc. The executions at the network edge do not prevent from leveraging cloud services: in our example, for instance, the execution taking care of the emergency scene is reporting to the traffic monitoring system about the current situation regardless of whether the infrastructure connection is available or not. The system can then provide further information to the families of victims.

Clearly, similar functionality might be achieved with conventional mobile applications and – to some extent – with cloud-based services. However, these alternative solutions fall short of providing adequate security and functional safety guarantees. Cloud-based coordination, for example, might simply be too unreliable and slow for real-time coordination, especially in sparsely populated and low network capacity areas with sporadic demand (e.g., traffic jam caused by the accident). Purely mobile applications, on the other hand, do not have such support for coordination by design. Additionally, this would require inclusion of coalition forming capabilities into each enabled application and/or involvement of the authorities, so that the established coalitions could be made secure and trusted. It would substantially complicate the development process. Fortunately, AcOP makes it simple to develop coordination that takes place at the network edge, which is much less prone to the erratic connections and can even operate without any connection to the outside world. Further pain points of the traditional approaches are summarized in Table 1.



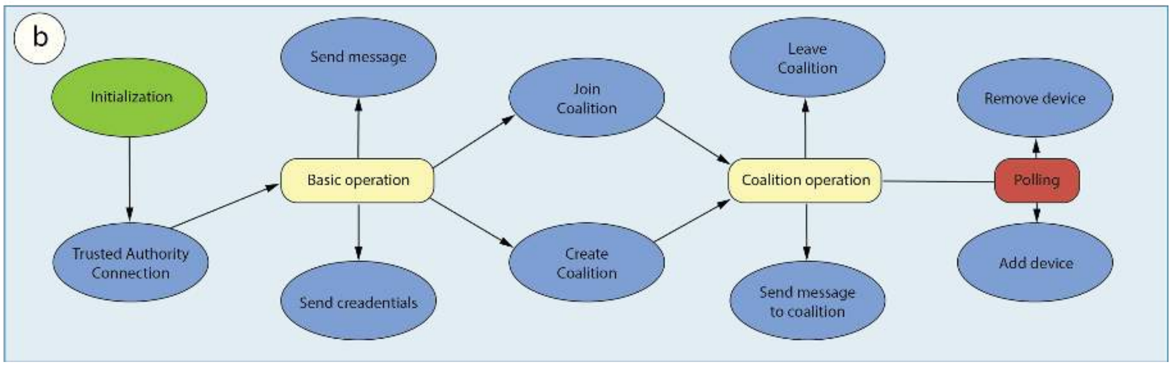


Figure 1.a: Example operation of AcOP model in an emergency scenario: User and device states (yellow and orange) are reported to collective executions (light blue). Actions (purple) are coordinating the operation of various devices with capabilities (green). One device at a time acts as coordinator, but the same app is executed collectively by multiple devices.
Figure1.b: The structure of the coalitions operating behind the collective executions of AcOP.

Table 1: Comparing mobile app (M) and cloud service based (C) approaches with our AcOP model (A) for CPS development.

M	• Easy to catch the application's errors with try-catch notation. • Some recovery can be tried, but typically the app crashes which does not support functional safety. • *If the device starts malfunctioning, how to support functional safety and replace it with another one?* • *How to handle errors that occur in the coordination or on other devices?*	Stand-alone, No functional safety
C	• Easy to catch errors that occur on the server side, and some contingencies on the device side can be sent to the server side and solved there. • Devices may be replaced with another device if such device is connected, the device owns desired capabilities, and the device is located so that it makes sense to use the device. • *How to manage errors in between the cloud and devices if Internet connection is lost, or how to fix poorly functioning connections?*	Multi-device, Limited functional safety
A	• AcOP has *contingency handlers* for remote and coordination related issues. Within the handler methods, developers can define ways for recovering from unwanted behavior, e.g., replace a disconnected or malfunctioning device with another one, change the connectivity type, etc. • It is also possible to reschedule an action, or take a completely different action.	Multi-device, Improved functional safety

M	• Easy to implement detection, e.g., when device location and/or orientation change (for instance, with the use of delegate methods). • *How to detect state changes on other devices?*	Single-source, Privacy by Design
C	• Data coming from multiple devices and sensors can be streamed to the cloud services and processed there when Internet connectivity is available. • *How does this approach scale when there are thousands of devices streaming data continuously and the Internet connectivity is poor quality?* • *Can the provider be trusted so that it does not save any data and use it without the user knowing?*	Multi-source, Privacy by Trust
A	• Collective execution is designed to be used for detecting sensation on a device that is participating in the same execution. Then, for the task at hand, the sensations can be combined and processed from as many sources as required. • The sensations are only shared among the devices participating to the same execution, which only contain information relevant for the execution. The data vanishes when the "on-the-fly" computation ends.	Multi-source, Privacy by Design

M	• When a sensation has been detected, the device can be instructed to act upon this event. • *How to select and command another device or a group of devices to act?*	Internet not required, No functional safety
C	• Cloud-based approaches are typically used for coordinating the CPS devices. The coordination, however, relies on the Internet connectivity, and communication in latency-sensitive systems can easily become a bottleneck. • *How to ensure real-time coordination without adequate Internet connectivity?* • *How to operate and coordinate independently if there is no Internet connectivity?*	Internet required, No functional safety
A	• In a collective execution on the network edge, a device detects certain sensation(s) and attempts to schedule an action. Actions are designed to serve as the output to the world, for commanding joint operations between one or many devices when performing a certain task. • Requires connectivity, but the connectivity does not have to be good quality Internet connection. • The trusted entities can be used for replacing one device with another to support functional safety.	Internet not required, Functional safety

M	• Particular tasks can be posted to be executed by the background cloud service. For instance, recent serverless approaches (e.g., Google's Cloud Functions, Azure Functions, and AWS Lambda) gained certain popularity. While the developer is liberated from server management, the computation distribution still takes place on the servers [8].	N/A
C	• The executions in cloud-based approaches can be either centralized or distributed when executed by several services and machines [9]. • *How to harness other (nearby) devices to perform computation in a safe and secure manner?*	Internet required, Privacy and security not guaranteed
A	• Collective executions and actions can run on any of the edge devices capable of handling JavaScript. It is natural to motivate such distribution: privacy and security of content and data to be utilized on a specific device. • In the collective executions, the coordination happens among trusted entities of the coalition.	Internet not required, Privacy and security guaranteed

M	• Mobile apps are deployed onto the devices via application stores. The cooperation requires that the other devices must have the same app installed and running. • *How to deploy cooperation in a flexible and safe manner, and without manual user input?*	Static, No constant Internet connection
C	• Tasks can be deployed onto cloud services, where the devices are instructed to cooperate. This requires constant connection between the devices and cloud in order to coordinate the operations. • *How to deploy a certain well-defined task to be executed by particular devices in all circumstances?*	Dynamic, Constant Internet connection
A	• AcOP components for executing certain tasks can be downloaded/installed from a repository in advance, or dynamically acquired at runtime whenever needed. • It is enough if one of the participating devices has preloaded the AcOP components that are taking care of the coordination. Hence on a zone where is no Internet connection, the device with the latest version of the components gets selected to the role of the coordinator.	Dynamic, No constant Internet connection

3 Establishing Secure Communicating Coalitions

To liberate the programmers from considering coalition formation as part of the application logic, an appropriate framework is required to enable operation of communicating peripherals, e.g., in case of a traffic accident described in the previous section. Certain known approaches exist already, thus bringing attention to the challenge of sandboxed executions in the emerging CPSs (e.g., FlowFence [10]). However, sandboxed collective executions of the same piece of software on the edge devices have not gained sufficient attention thus far.

Today, mobile devices may establish and utilize a direct link only if they have a reliable connection to the coordinator, which is responsible for the secure connection management, or if they trigger the connection themselves. In the latter case, no security and safety guarantees can be provided by the operator. To mitigate this limitation, Public Key Infrastructure is commonly utilized for enabling secure and authenticated communication when a connection to the centralized authority is available [11]. Without it, many applications might become disabled if but a single user leaves the network coverage. This particularly occurs in cases of disaster and/or when a cellular connection is unreliable (network is overloaded) or unavailable (train, airplane, elevator, etc.).

To augment the edge computing technologies, we propose to employ a secure communication framework that we developed in a series of trials in a live cellular network [12]. Our system is built upon the advanced security protocols contributed by 3GPP specifications. This novel framework applies the knowledge of distributed solutions to enable secure communication. Accordingly, execution in people's devices in the emergency scene enables them to seamlessly join and leave a coalition without disrupting collective execution.

The main operation phases of the considered approach are illustrated in Fig. 1.b. The only procedure that requires stable connectivity to the cloud is the coalition initialization. First, the involved mobile devices receive their certificates with the corresponding secret and public keys. These are utilized to establish secure direct connectivity with each relevant device. When a device is willing to create a secure coalition with its "neighbors", a request containing the public identifiers of future coalition members is sent to the corresponding server. The coalition secret is then generated and split between the coalition users.

A polling procedure is then triggered by the network to ensure that the subject devices are actually willing to join this coalition. After the confirmations have been received, both coalition certificate and coalition secret (based on the use of Lagrange polynomials) are delivered. After these steps, secure direct interaction may continue over any conventional network. The members of an existing coalition have the possibility to invite new devices as well as remove the existing ones based on the flexible voting system, i.e., when k out of M coalition members agree on a particular decision (runs automatically for machines and can be manual for humans). This allows the coalition to be updated dynamically to manage collective executions in various scenarios. For example, predefined "hidden" coalitions may be utilized in cases of disaster, thus enabling operational stability.

4 Summary

Edge computing is increasingly demanded due to the CPS requirements for increased scalability and functional safety – if the entities are coordinated by the cloud, a risk remains that without reliable Internet connectivity the functional safety cannot be guaranteed. In cooperation at the network edges, devices need to be able to trust each other, thus calling for dynamic coalitions with secure and trusted topology. This, in its turn, improves functional safety since trusted entities can cooperate and act as back-up options for one another in various CPS applications (see Fig. 2): if one device fails, others are there to stand in. In order to achieve this, we proposed an Action-Oriented Programming model and the associated framework that can dynamically adapt to the edge and the cloud.

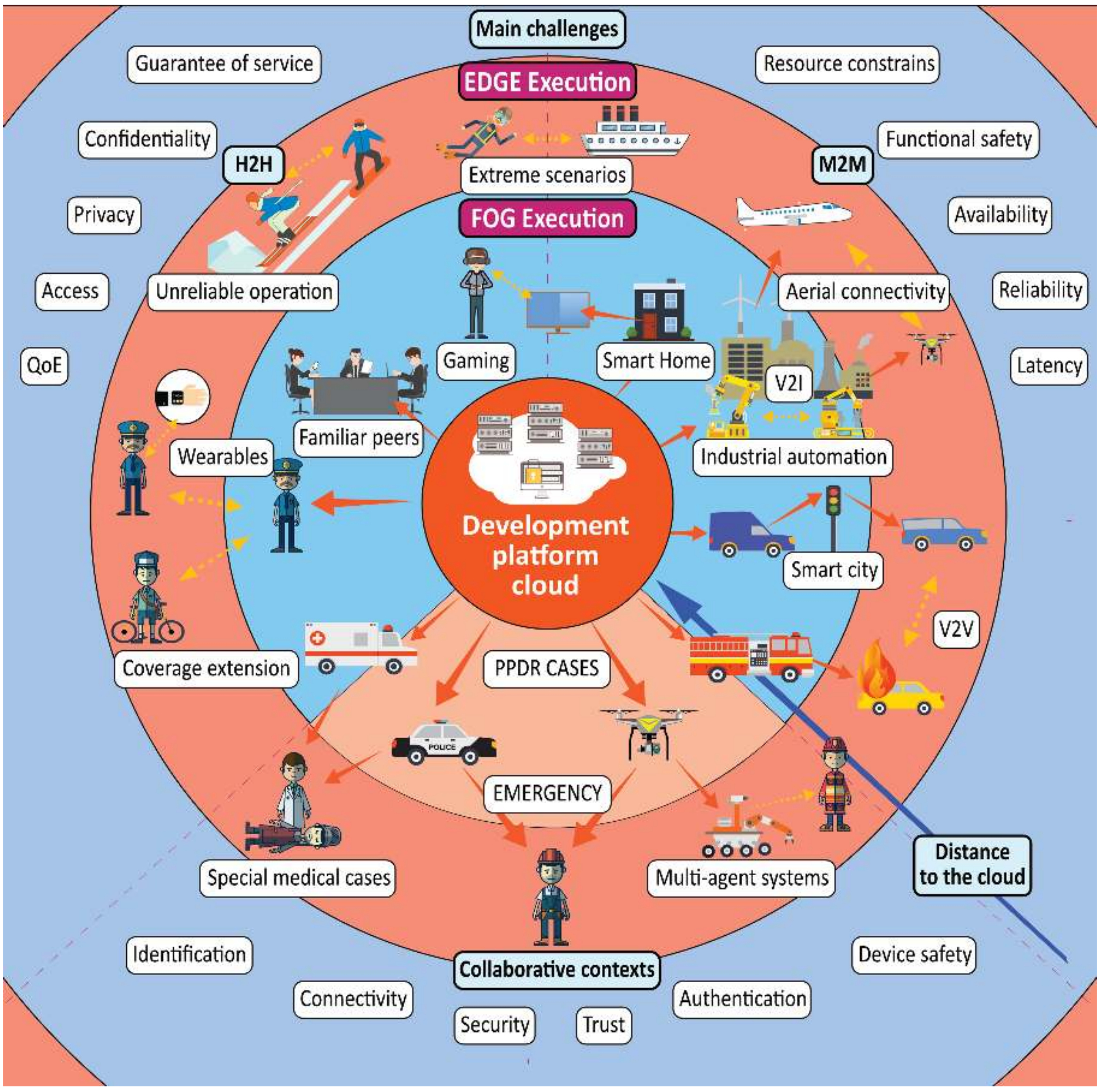


Figure 2: Considered application scenarios of cloud, fog, and edge computing for Action-Oriented Programming model

References

[1] P. J. Mosterman and J. Zander, "Cyber-physical systems challenges: a needs analysis for collaborating embedded software systems," *Software & Systems Modeling*, vol. 15, no. 1, pp. 5–16, 2016.

[2] E. Elmroth, P. Leitner, S. Schulte, and S. Venugopal, "Connecting Fog and Cloud Computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 22–25, 2017.

[3] R. Tandon and O. Simeone, "Harnessing Cloud and Edge Synergies: Toward an Information Theory of Fog Radio Access Networks," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 44–50, 2016.

[4] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, 2016.

[5] M. Swan, "The Quantified Self: Fundamental Disruption in Big Data Science and Biological Discovery," *Big Data*, vol. 1, no. 2, pp. 85–99, 2013.

[6] N. Mäkitalo, J. Pääkkö, M. Raatikainen, V. Myllärniemi, T. Aaltonen, T. Leppänen, T. Männistö, and T. Mikkonen, "Social Devices: Collaborative Co-located Interactions in a Mobile Cloud," in *Proc. of the 11th International Conference on Mobile and Ubiquitous Multimedia*, p. 10, ACM, 2012.

[7] A. Mashkoor and M. Biro, "Towards the Trustworthy Development of Active Medical Devices: A Hemodialysis Case Study," *IEEE Embedded Systems Letters*, vol. 8, no. 1, pp. 14–17, 2016.

[8] A. Eivy, "Be Wary of the Economics of "Serverless" Cloud Computing," *IEEE Cloud Computing*, vol. 4, pp. 6–12, 2017.

[9] Y. Zhao, K. Yoshigoe, M. Xie, S. Zhou, R. Seker, and J. Bian, "Evaluation and Analysis of Distributed Graph-Parallel Processing Frameworks," *Journal of Cyber Security and Mobility*, vol. 3, no. 3, pp. 289–316, 2014.

[10] E. Fernandes, J. Paupore, A. Rahmati, D. Simionato, M. Conti, and A. Prakash, "FlowFence: Practical Data Protection for Emerging IoT Application Frameworks," in *Proc. of USENIX Security Symposium*, 2016.

[11] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Professional, 2003.

[12] A. Ometov, P. Masek, J. Urama, J. Hosek, S. Andreev, and Y. Koucheryavy, "Implementing secure network-assisted D2D framework in live 3GPP LTE deployment," in *Proc. of International Conference on Communications Workshops (ICC)*, IEEE, pp. 749-754, 2016.