Modeling of Sensor Clouds Under the Sensing as a Service Paradigm

J. Guerreiro(✉), L. Rodrigues, and N. Correia

CEOT, FCT, University of Algarve, 8005-139 Faro, Portugal
{jdguerreiro,lrodrig,ncorreia}@ualg.pt

Abstract. 5G technologies will facilitate the emergence of applications integrating multiple physical Things. In such scenarios, Cloud-integrated platforms end up having a key role due to their storage and processing capabilities. Therefore, a clear understanding of Sensor Clouds, and on how Cloud mechanisms can be orchestrated to better face requests, becomes a very relevant issue as Sensing as a Service models emerge. This article presents a model for Sensor Clouds, suitable for emerging IoT related Sensing as a Service business models. Such a model is used to assess the impact of resource allocation approaches and unveil the trade-off between scalability, elasticity and quality of experience. Results show that the best resource allocation approach is highly dependent on the suppliers/consumers scenario.

1 Introduction

With the *Internet of Things* (IoT), devices/Things can easily exchange data over the Internet. Resources can be easily discovered, accessed and managed, making Things accessible to a large pool of developers. The 5th generation wireless technology will have a key role in such scenarios and 5G IoT is already called the Internet of everyone and everything [1]. 5G technologies meet the requirements of mobile communications and needs for Thing data transmission, facilitating the emergence of applications integrating multiple physical Things with virtual resources available at the Internet/Web. The *Sensing as a Service* (Se-aaS) model emerges from this reality, allowing everyone to benefit from such an IoT eco-system, and Cloud-integrated platforms are usually used due to their storage and processing capabilities [2,3].

The most relevant "as a service" models under the Cloud Computing paradigm are: (*i*) *Infrastructure as a Service* (IaaS), providing computing resources (e.g., virtual machines); (*ii*) *Platform as a Service* (PaaS), providing computing platforms that may include an operating system, database, Web server, and others; (*iii*) *Software as a Service* (SaaS), where the Cloud takes over the infrastructure and platform while scaling automatically [4]. The Se-aaS model emerged more recently for sensors/data to be shared. This means that there is a multi-supplier deployment of sensors, and multi-client access to resources [5]. Cloud service providers should find some incentive mechanism for device owners to participate [6,7].

V. Sucasas et al. (Eds.): BROADNETS 2018, LNICST 263, pp. 22–30, 2019.
https://doi.org/10.1007/978-3-030-05195-2_3

In this article, the modeling of Sensor Clouds is addressed. Although a first attempt was done in [5], their focus is on *Wireless Mesh Networks* (WSNs) and on how these can move to Sensor Clouds, not being adequate for other IoT Se-aaS business models. More specifically, in [5] sensors are allocated to a single application and mashups are not addressed. Therefore, it can be seen as a WSN virtualization. Here we propose a model that is suitable for emerging IoT related Se-aaS business models, including the one in [5]. This model considers sensors/data sharing by multiple applications and mashups, allowing one to assess the impact of resource allocation approaches (both Cloud and physical Things), and better understanding of trade-offs, so that mechanisms can be orchestrated to face future requests.

The remainder of this article is organized as follows. In Sect. 2, previous research on Se-aaS paradigm is discussed. Section 3 presents the mathematical model, while Sect. 4 analyses it and discusses trade-offs. Finally, Sect. 5 draws conclusions.

2 Related Work

The Se-aaS was initially discussed in [6] for Cloud-based sensing services using mobile phones. Such work analyses its design and implementation challenges. In the context of smart cities, Se-aaS is discussed in [2,8]. The first addresses technological, economical and social perspectives, while the last proposes the abstraction of physical objects through semantics, so that devices can be integrated by neglecting their underlying architecture. In [9,10], the semantic selection of sensors is also addressed.

Multimedia Se-aaS is explored in [11–13]. These focus on real-type communication requirements, and [13] explores Cloud edges and fogs. For a survey on mobile multimedia Cloud computing see [14].

In [5,15,16], the integration of WSNs with the Cloud is investigated. Their concerns are mainly data storage and/or device assignment to tasks. In [5], a WSN virtualization model is discussed.

This article, contrarily to previous works, addresses a multi-supplier and multi-client modeling of Sensor Clouds, allowing client applications to request for available devices and build mashups. The focus is not on crowd sensing making data from mobile phones available to multiple clients, but instead on how applications can share devices and build mashups, which is not considered in [5,15,16]. Mashups may lead to internal data flows in the Cloud, if data is used by different mashups (each integrating it differently with other Internet/Web resources), and this has not been accounted for by previous approaches.

3 Theoretical Modeling of Sensor Cloud

3.1 Definitions

Definition 1 (Physical Thing). *A sensor detecting events/changes, or an actuator receiving commands for the control of a mechanism/system. The model of a physical Thing i includes all properties necessary to describe it, denoted by*

$\mathcal{P}_i$, and all its functionalities, denoted by $\mathcal{F}_i$. That is, $\mathcal{P}_i = \{p : p \in \mathcal{P}\}$, where $\mathcal{P}$ is the overall set of properties (e.g., sensing range, communication facility, location), and $\mathcal{F}_i = \{f : f \in \mathcal{F}\}$, where $\mathcal{F}$ is the overall set of functionalities (e.g., image sensor), considering all devices registered at the Cloud.

It is assumed that properties and functionalities, at $\mathcal{P}$ and $\mathcal{F}$ respectively, result from a semantic description of physical Things registered at the Cloud. That is, specific vocabularies are used when naming properties and functionalities (see [17], for example). Each property $p_i \in \mathcal{P}_i$ has a "subject/predicate/object" description[1] denoted by $spo(p_i)$ (e.g., temperature hasValue 30 °C). The set of all physical Things is denoted by $\mathcal{T}^P$, and sensor owners voluntarily register/deregister physical Things to/from the Cloud.

Definition 2 (Mashup). *Workflow built by wiring together Things and services from various Web sources, on which an application is based.*

That is, applications (at the user side) should be able to access Things at the Cloud and, if necessary, blend them with other services and data sources on the Web, as shown in Fig. 1. However, for resources to be used efficiently, applications should not pick physical Things directly. Instead, a functionality requirement and minimum/maximum property requirements should be specified for each element n included in a mashup, denoted by $\bar{f}_n$ and $\bar{\mathcal{P}}_n$, allowing then an optimized allocation of physical Things to mashup elements. Each $p_n \in \bar{\mathcal{P}}_n$ can have a "subject/predicate/object" description of the condition/requirement that is being defined (e.g., cameraResolution greaterThan 12.1MP; frequencySampling equalTo 10 s), denoted by $spo(p_n)$. The overall population of mashup elements (from all applications) at the Cloud will be denoted by $\mathcal{N}$.

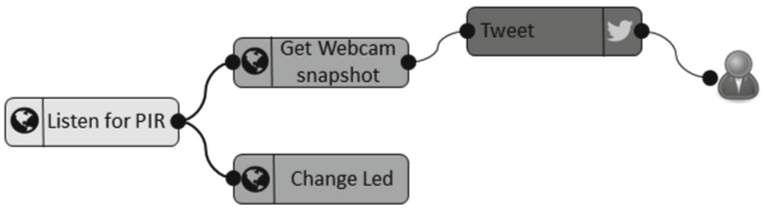

Fig. 1. Thing mashup.

For devices/data to be consumed by multiple applications, virtual Things will be created at the Cloud. Then, each mashup element is binded to a single virtual Thing, while a virtual Thing can be binded to multiple mashup elements (with same functionality and compatible property requirements). Basically, virtual Things represent multiple mashup elements, from multiple applications,

[1] A Resource Description Framework (RDF) triple.

and these are the ones to be materialized into physical Things. Such an approach allows data generated by a virtual Thing to be consumed by multiple applications, while reducing data collection/storage and increasing data utility. Mashup elements are, however, application dependent.

Definition 3 (Virtual Thing). *Thing at the Cloud to which mashup elements are binded to. A virtual Thing j is materialized through one or more concerted physical Things, denoted by $\mathcal{M}_j$, $\mathcal{M}_j \subset \mathcal{T}^P$, able to provide the requirements associated with the virtual Thing (requirements from all mashup elements binded to it). Therefore, $f_j \triangleq \cup_{i \in \mathcal{M}_j} \mathcal{F}_i$ and $\mathcal{P}_j = \cup_{i \in \mathcal{M}_j} \mathcal{P}_i$.*

That is, a virtual Thing can have one or multiple physical Things in the background working together. The set of all virtual Things is denoted by $\mathcal{T}^V$.

With virtualization users remain unaware of the physical devices used, allowing these to be dynamically allocated to virtual Things. The client ends up having no deployment and maintenance costs, while having an on-demand fault tolerant service because virtual Things can always use other available physical Things.

3.2 The Model

Assumptions: A *Cloud Service Provider* (CSP), denoted by $\mathcal{S}$, includes a set of distributed computing resources, each set serving a certain region or having a certain role. Therefore, $\mathcal{S} = \{\mathcal{S}_1, ...\mathcal{S}_{|\mathcal{S}|}\}$. The set of applications (outside the Cloud), requesting for sensors with certain properties, is denoted by $\mathcal{A} = \{\mathcal{A}_1, ..., \mathcal{A}_{|\mathcal{A}|}\}$. An application $\mathcal{A}_i$ can have one or more independent components, denoted by $\mathcal{C}(\mathcal{A}_i) = \{\mathcal{C}_1^i, ...\mathcal{C}_{|\mathcal{C}(\mathcal{A}_i)|}^i\}$, and each component $\mathcal{C}_j^i$ is binded to a mashup (at the Cloud) of δ_j^i steps, $\mathcal{C}_j^i \triangleq \{1, ..., \delta_j^i\}$. The following is also assumed:

- Web templates are used to draw the mashup associated with each component, where minimum/maximum property and functionality requirements are specified for each mashup element. Elements can be connected, and $succ(n)$ denotes the successors of element n at the mashup workflow (elements to which n sends data to).
- Final mashups' data is sent to the corresponding application components through bindings.
- Virtual Things are created, and binded to mashup elements, by the Cloud.

Formalization: One or more physical Things materialize one virtual Thing. Assuming $\tau^i = \{\mathcal{T}_1^{P,i}, \mathcal{T}_2^{P,i}, ...\}$ to be a partition of $\mathcal{T}^P$, function $g_1 : \tau^i \rightarrow \mathcal{T}^V$ is defined for virtual Thing materialization:

$$g_1(\mathcal{T}_j^{P,i}) = \{\exists! k \in \mathcal{T}^V : f_k \triangleq \cup_{l \in \mathcal{T}_j^{P,i}} \mathcal{F}_l\}. \quad (1)$$

This states that a virtual Thing $k \in \mathcal{T}^V$ is mapped to $\mathcal{T}_j^{P,i}$ if they are functionally similar. Assuming now $\eta^i = \{\mathcal{N}_1^i, \mathcal{N}_2^i, ...\}$ to be a partition of $\mathcal{N}$ (all elements in $\mathcal{N}_j^i$ with the same functionality requirement), function $g_2 : \eta \rightarrow \mathcal{T}^V$ is defined to bind $\mathcal{N}_j^i$ to a virtual Thing:

$$g_2(\mathcal{N}_j^i) = \{\exists! k \in \mathcal{T}^V : \bar{f}_n = \mathcal{F}_k \wedge \bar{\mathcal{P}}_n \subseteq \mathcal{P}_k \wedge \Delta(spo(p_n), spo(p_k)) = \text{true}, \\ , \forall n \in \mathcal{N}_j^i, \forall p_n \in \bar{\mathcal{P}}_n, \forall p_k \in \mathcal{P}_k\}. \quad (2)$$

where Δ specifies whether p_n is compatible with p_k, or not. This states that a virtual Thing $k \in \mathcal{T}^V$ mapped to $\mathcal{N}_j^i$ must: (i) provide the functionality being requested by elements in $\mathcal{N}_j^i$; (ii) fulfill the property requirements of all elements in $\mathcal{N}_j^i$.

Different resource allocation approaches (partitions and allocations done by g_1 and g_2) can be adopted by sensor Clouds, each with an impact on scalability, elasticity and *Quality of Experience* (QoE). Let us assume that η^U is the universe set of all feasible partitions of mashup elements, $\eta^U = \{\eta^1, \eta^2, ..., \eta^{|\eta^U|}\}$ and $\eta^i = \{\mathcal{N}_1^i, \mathcal{N}_2^i, ..., \mathcal{N}_{|\mathcal{T}^V|}^i\}$, $\forall i \in \{1, ..., |\eta^U|\}$. Also, τ^U is the universe set of all feasible partitions of physical Things, $\tau^U = \{\tau^1, \tau^2, ..., \tau^{|\tau^U|}\}$ and $\tau^i = \{\mathcal{T}_1^{P,i}, \mathcal{T}_2^{P,i}, ..., \mathcal{T}_{|\mathcal{T}^V|}^{P,i}\}$, $\forall i \in \{1, ..., |\tau^U|\}$. Thus, each element in τ^U is a feasible materialization of virtual Things. For such universe sets, the most scalable resource allocation approach (system can accommodate more load/clients in the future) would select the following solution:

$$(\eta^i, \tau^j)^{SCA} = argmin_{\eta^i \in \eta^U}\{|\eta^i|\}. \quad (3)$$

That is, since each element of partition η^i will be associated with a virtual Thing, fewer virtual Things not only means less virtual workspaces but also more productive virtual Things, as data flowing from them serves more mashups/applications.

Elasticity is the ability to adapt resources to loads. That is, resources should become available when the load increases, but when the load decreases then unneeded resources should be released. Thus, the most elastic resource allocation approach would select the following solution:

$$(\eta^i, \tau^j)^{ELA} = argmin_{\eta^i \in \eta^U}\{max_{\mathcal{S}_l \in \mathcal{S}}\{\sum_{k \in \mathcal{T}^V} \xi(k, \mathcal{S}_l)\}\} \quad (4)$$

where $\xi(k, \mathcal{S}_l)$ is the amount of computational resources allocated to virtual Thing k at $\mathcal{S}_l$. Therefore, virtual Things are evenly distributed by CSPs.

Regarding the resource allocation approach with a better impact on the QoE perceived by users, this would be the one selecting the following solution:

$$(\eta^i, \tau^j)^{QoE} = argmin_{\eta^i \in \eta^U, \tau^j \in \tau^U}\{h(\eta^i, \tau^j)\} \quad (5)$$

where $h : \eta^U \times \tau^U \rightarrow \Re^+$ is a cost function defined as:

$$h(\eta^i, \tau^j) = \sum_{k \in \mathcal{T}^V} \sum_{k' \in \chi(k)} TF^{V2V}(k, k') + \sum_{\mathcal{A}_i \in \mathcal{A}} \sum_{k \in \Phi(\mathcal{A}_i)} TF^{V2A}(k, \mathcal{A}_i) + \\ + \sum_{k \in \mathcal{T}^V} \sum_{k' \in \Psi(k)} TF^{P2V}(k', k). \quad (6)$$

The $TF^{V2V}(k, k')$ is a transfer cost associated with the data flow between the workspaces of virtual Things k and k' at the Cloud (Virtual-to-Virtual cost), because mashup elements can be connected. The $\chi(k)$ must provide all virtual Things requiring data flow from virtual Thing k. That is,

$$\chi(k) = \{k'' \in \mathcal{T}^V : k = g_2(\mathcal{N}_l^i), k'' = g_2(\mathcal{N}_m^i) \wedge \ n' = succ(n), n \in \mathcal{N}_l^i, \\ , n' \in \mathcal{N}_m^i, \mathcal{N}_l^i, \mathcal{N}_m^i \in \eta^i\}. \quad (7)$$

The $TF^{V2A}(k, \mathcal{A}_i)$ is a transfer cost associated with the data flow between the workspace of virtual Thing k and the user application $\mathcal{A}_i$. The $\Phi(\mathcal{A}_i)$ provides all virtual Things consumed by application $\mathcal{A}_i$,

$$\Phi(\mathcal{A}_i) = \{k' \in \mathcal{T}^V : k' = g_2(\mathcal{N}_l^i) \wedge succ(n) = \emptyset \wedge n \in \mathcal{C}_j^i, \mathcal{N}_l^i \in \eta^i, n \in \mathcal{N}_l^i, \\ , \mathcal{C}_j^i \in \mathcal{C}(\mathcal{A}_i)\}. \quad (8)$$

Finally, the $TF^{P2V}(k', k)$ is a transfer cost associated with the data flow between the physical Thing k' (or its corresponding proxy/gateway) and the workspace of virtual Thing k, which depends on the materialization of k. Therefore, $\Psi(k)$ will be

$$\Psi(k) = \{k'' \in \mathcal{T}^P : k = g_1(\mathcal{T}_j^{P,i}) \wedge k'' \in \mathcal{M}_k, \mathcal{T}_j^{P,i} \in \tau^j\}. \quad (9)$$

Regarding the transfer cost itself, this may include the number of hops, processing required at the destination, etc, or any combination of these.

4 Analysis of Results

4.1 Scenario Setup

A set of random graphs, using the algorithm in [18], were used to apply the model described. These graphs, each with 10 nodes, represent the location of CSP's resources, $\mathcal{S}_1, ...\mathcal{S}_{|\mathcal{S}|}$. There are $|\mathcal{A}| = \kappa_1 \times |\mathcal{S}|$ applications and $|\mathcal{T}^P| = \kappa_2 \times |\mathcal{S}|$ physical Things registered at the Sensor Cloud, where κ_1 and κ_2 are integers. Each $\mathcal{S}_i \in \mathcal{S}$ connects, on average, $\frac{|\mathcal{A}|}{|\mathcal{S}|}$ applications and $\frac{|\mathcal{T}^P|}{|\mathcal{S}|}$ physical Things to the Cloud.

The virtual Things to be built depend on physical Things, application requirements and aggregation level when allocating mashup elements to virtual Things. Therefore, tests were done for different amounts of virtual Things,

$\frac{|\mathcal{A}|\times\kappa_3\times\kappa_4}{10} \leq |\mathcal{T}^V| \leq \frac{|\mathcal{A}|\times\kappa_3\times\kappa_4}{2}$, where κ_3 is the average number of components per application and κ_4 is the average number of elements at mashups. For transfer costs in Eq. (6), the following is assumed:

- TF^{V2A}: Since there will be κ_3 bindings of data flow from the Cloud to an application, a virtual Thing k will send its data towards application $\mathcal{A}_i$ with probability $prob(k, \mathcal{A}_i) = \frac{\kappa_3}{|\mathcal{T}^V|}$.
- TF^{V2V}: Since each mashup has $\kappa_4 - 1$ flow links[2], a virtual Thing k has a data flow towards k' with probability $prob(k, k') = \frac{(\kappa_4-1)\times\kappa_3\times|\mathcal{A}|}{|\mathcal{T}^V|\times(|\mathcal{T}^V|-1)} \times \alpha$, where α is the virtual Thing sharing factor or ratio $\frac{\kappa_4\times\kappa_3\times|\mathcal{A}|}{|\mathcal{T}^V|}$.
- TF^{P2V}: A physical Thing k' has a data flow towards virtual Thing k with probability $prob(k', k) = \frac{\kappa_5}{|\mathcal{T}^P|}$, where κ_5 is the average number of physical Things in a virtual Thing materialization.
- The number of hops is assumed to be the transfer cost in TF^{V2A}, TF^{V2V} and TF^{P2V}.

Table 1 shows the parameter values assumed.

Table 1. Simulation parameters.

Parameter	Value
Number of nodes at CSP graph ($\|\mathcal{S}\|$)	10
Number of applications ($\|\mathcal{A}\|$)	30
Number of physical Things ($\|\mathcal{T}^P\|$)	30
Avg number of components per app (κ_3)	3
Avg number of elements at each mashup (κ_4)	3
Virtual Thing materialization factor (κ_5)	1
Lowest number of virtual Things	$\frac{\|\mathcal{A}\|\times\kappa_3\times\kappa_4}{10}$
Highest number of virtual Things	$\frac{\|\mathcal{A}\|\times\kappa_3\times\kappa_4}{2}$

4.2 Discussion

Figure 2 shows[3] the impact of $|\eta^i|$ (or number of virtual Things), which is a consequence of the aggregation level used by resource allocation approaches. Less virtual Things means that solutions are more scalable.

A relevant observation regarding the impact of making more or less scalable choices (virtual Things serving more or less applications), is that in general the QoE and elasticity improve as Sensor Clouds choose for less scalable solutions. In this case, virtual Things are serving less applications and, therefore, less data transfers between virtual Things occurs and data takes less hops to flow towards

[2] A flow tree is assumed.

[3] Average of results obtained for all generated graphs.

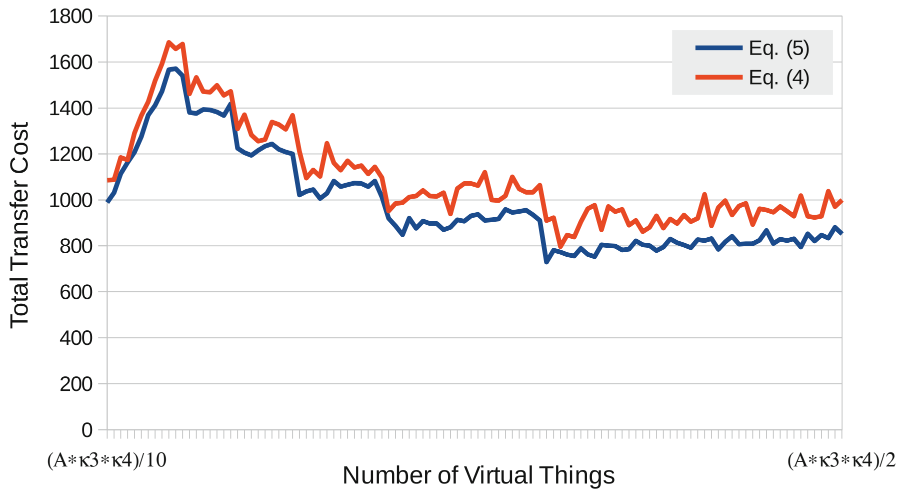

Fig. 2. Impact of $|\eta^i|$ (or number of virtual Things).

applications. Also, for each virtual Thing there will be less load. However, this does not happen for a small number of virtual Things. In this case, increasing the number of virtual Things leads to a higher transfer cost, with a negative impact on QoE, and worse elasticity. This happens because virtual Things are already highly dependent, and flow from physical Things towards the Cloud takes over the previously mentioned benefit of using more virtual Things. Thus, scalability can have a positive or negative impact on QoE and elasticity depending on the scenario (mashups, etc), which will determine possible allocations of mashup elements to virtual Things, and the best resource allocation approach to use.

5 Conclusions

In this article, a model for Sensor Clouds is presented allowing the impact of resource allocation approaches to be assessed, and trade-off between scalability and QoE/elasticity to be unveiled. Results also show that the best resource allocation approach to use will depend on mashups, etc, influencing possible allocations of mashup elements to virtual Things. This awareness allows Sensor Cloud providers to choose the best approach according to their case.

Acknowledgment. This work was supported by FCT (Foundation for Science and Technology) from Portugal within CEOT (Center for Electronic, Optoelectronic and Telecommunications) and UID/MULTI/00631/2013 project.

References

1. Zhang, D., Zhou, Z., Mumtaz, S., Rodriguez, J., Sato, T.: One integrated energy efficiency proposal for 5G IoT communications. IEEE Internet Things J. **3**(6), 1346–1354 (2016)
2. Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D.: Sensing as a service model for smart cities supported by Internet of Things. In: Transactions on Emerging Telecommunications Technologies, vol. 25, No. 1. John Wiley & Sons, Inc. New York (2014)
3. Perera, C., Talagala, D.S., Liu, C.H., Estrella, J.C.: Energy-efficient location and activity-aware on-demand mobile distributed sensing platform for sensing as a service in IoT clouds. IEEE Trans. Comput. Soc. Syst. **2**(4), 171–181 (2015)
4. Duan, Y., et al.: Everything as a service (XaaS) on the cloud: origins, current and future trends. IEEE International Conference on CLOUD (2015)
5. Misra, S., Chatterjee, S., Obaidat, M.S.: On theoretical modeling of sensor cloud: a paradigm shift from wireless sensor network. IEEE Syst. J. **PP**(99) (2014)
6. Sheng, X., Tang, J., Xiao, X., Xue, G.: Sensing as a service: challenges, solutions and future directions. IEEE Sens. J. **13**(10), 3733–3741 (2013)
7. Pouryazdan, M., Kantarci, B., Soyata, T., Foschini, L., Song, H.: Quantifying user reputation scores, data trustworthiness, and user incentives in mobile crowdsensing. IEEE Access **PP**(99) (2017)
8. Petrolo, R., Loscrì, V., Mitton, N.: Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. Transactions on Emerging Telecommunications Technologies, Wiley Online (2015)
9. Misra, S., Bera, S., Mondal, A., Tirkey, R., Chao, H.-C., Chattopadhyay, S.: Optimal gateway selection in sensor-cloud framework for health monitoring. IET Wireless Sens. Syst. **4**(2), 61–68 (2014)
10. Hsu, Y.-C., Lin, C.-H., Chen, W.-T.: Design of a sensing service architecture for internet of things with semantic sensor selection. In: International Conference on UTC-ATC-ScalCom (2014)
11. Lai, C.-F., Chao, H.-C., Lai, Y.-X., Wan, J.: Cloud-assisted real-time transrating for HTTP live streaming. IEEE Wireless Commun. **20**(3), 62–70 (2013)
12. Lai, C.-F., Wang, H., Chao, H.-C., Nan, G.: A network and device aware QoS approach for cloud-based mobile streaming. IEEE Trans. Multimedia **15**(4), 747–757 (2013)
13. Wang, W., Wang, Q., Sohraby, K.: Multimedia sensing as a service (MSaaS): exploring resource saving potentials of at cloud-edge IoTs and fogs. IEEE Internet Things J. **PP**(99) (2016)
14. Xu, Y., Mao, S.: A survey of mobile cloud computing for rich media applications. IEEE Wireless Commun. **20**(3), 46–53 (2013)
15. Zhu, C., Li, X., Ji, H., Leung, V.C.M.: Towards integration of wireless sensor networks and cloud computing. IEEE International Conference on CloudCom (2015)
16. Dinesh Kumar, L.P., et al.: Data filtering in wireless sensor networks using neural networks for storage in cloud. In: International Conference on ICRTIT (2012)
17. Compton, M., et al.: The SSN ontology of the W3C semantic sensor network incubator group. In: Web Semantics: Science, Services and Agents on the World Wide Web, Science Direct, Vol. 17. Elsevier (2012)
18. Onat, F.A., Stojmenovic, I.: Generating random graphs for wireless actuator networks. IEEE International Symposium WoWMoM (2007)