Trustworthy Opportunistic Access to the Internet of Services

Alessandro Armando, Aniello Castiglione, Gabriele Costa, Ugo Fiore, Alessio Merlo, Luca Verderame, Ilsun You

Trustworthy Opportunistic Access to the Internet of Services

Alessandro Armando[1], Aniello Castiglione[2], Gabriele Costa[1], Ugo Fiore[3], Alessio Merlo[14], Luca Verderame[1], and Ilsun You[5]

[1] Università degli Studi di Genova, Genova, Italy
name.surname@unige.it
[2] Università degli Studi di Salerno, Fisciano, Italy
castiglione@ieee.org
[3] Università di Napoli Federico II, Napoli, Italy
ufiore@unina.it
[4] Università E-Campus, Novedrate, Italy
alessio.merlo@uniecampus.it
[5] Korean Bible University, Seoul, Republic of Korea
isyou@bible.ac.kr

Abstract. Nowadays web services pervade the network experience of the users. Indeed, most of our activities over the internet consist in accessing remote services and interact with them. Clearly, this can happen only when two elements are available: (*i*) a compatible device and (*ii*) a suitable network connection. The recent improvement of the computational capabilities of mobile devices, e.g., tablets and smartphones, seriously mitigated the first aspect. Instead, the inappropriateness, or even the absence, of connectivity is still a major issue.
Although mobile, third generation (3G) networks can provide basic connectivity, complex interactions with web services often require different levels of Quality of Service (QoS). Also, 3G connectivity is only available in certain areas, e.g., user's country, and purchasing temporary connection abroad can be very costly. These costs weigh down on the original service price, seriously impacting the web service business model.
In this paper we describe the problems arising when considering the orchestration of service-oriented opportunistic networks and we present the assumptions that we want to consider in our context. We claim that our model is realistic mainly for two reasons: (*i*) we consider state-of-the-art technology and technical trends and (*ii*) we refer to a concrete problem for service providers.

1 Introduction

The evolution of Web 2.0 as well as the spread of cloud computing platforms have pushed customers to use always more remote services (hosted in a cloud or a server farm) rather than local ones (installed on personal devices). Such paradigm shift has basically improved the role of the network connectivity. Indeed, the access to remote services as well as the user experience, strongly depends on the network availability and the related performances (i.e., QoS).

To get evidence of this, let us consider a set of cloud users travelling through an airport and needing to access remote services from their device (e.g. smartphone, tablets, laptop) to complete their job. Presently, telecommunications companies sell internet connection for fixed time slots inside the airport, by means of 3G or wireless connections. Thus, each of these cloud users is compelled to subscribe, individually, to such internet connections, thereby getting extra charge to access the remote service. Moreover, users often do not get through the purchased connection, using less bandwidth or disconnecting before the end of the time slot. Then, such scenario leads to a non-negligible waste of purchased resources and money that may be reduced whether proper architectural or software solutions would allow, for instance, cooperation and resource sharing among the cloud users.

In this paper, we cope with such problem by investigating the adoption of the Software Defined Networking (SDN) paradigm as potential solution to build and manage QoS-constrained and on demand connection among mobile devices. In particular, we describe the main issues arising when trying to orchestrate a group of mobile devices that participate in an opportunistic network. Besides the difficulty of finding valid orchestration, e.g., in terms of QoS, we also present the security concerns at both network and device level. Finally we introduce a case study illustrating how our assumptions apply to a real life web service.

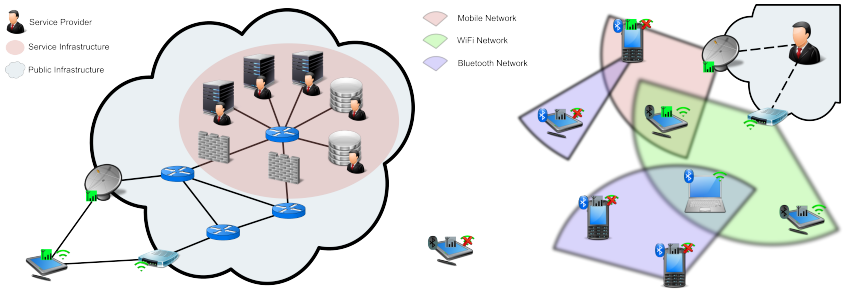
This paper is structured as follows. In Section 2 we state the problem of orchestrating opportunistic, service-oriented networks. Then, Section 3 describes the main security issues arising in this context and how to deal with them. In Section 4 we present our case study and its features. Finally, in Section 5 we survey on some related works and Section 6 concludes the paper.

2 Problem Statement

A provider P of a service S relies on a network infrastructure implementing S. The implementation of S is designed to meet both functional, e.g., accessibility, QoS and responsiveness, and non functional, e.g., security and fault tolerance, requirements. Moreover, through proper testing procedures, evidences that the implementation of S complies with these requirements have been produced and collected by P. In order to access S, customers need a network enabled device, e.g., laptops, tablets and smartphones, that can connect to and interact with S (typically by means of a client application). This scenario is depicted in Figure 1a.

Clearly, when a suitable connection is not available, the customer has no access to S. In order to access S, customers might enable a new connection, e.g., by buying a (costly and slow) 3G or a (local) wifi connection from a connectivity provider. This approach requires an existing infrastructure to be present and, definitely, charges extra costs on customers that, possibly, already pay for S.

Recent technological trends have highlighted that mobile devices can share their connectivity by playing the role of an access point. This technology, known as *tethering*, exploits multiple connection paradigms, e.g., wifi, bluetooth and



(a) Standard access configuration. (b) Service unreachable to customer.

Fig. 1. Mobile access to a web service.

IR, to create local networks. For instance, a mobile device can use wifi tethering to share its 3G connection with a group of neighbors. Although a single device has serious limitations, e.g., battery consumption, computational overhead and bandwidth exhaustion, populated areas are typically characterized by the presence of many devices. Thus, a proper orchestration of (a group of) these devices can lead to more powerful and reliable networks.

2.1 Local area configuration

A customer C having no network connectivity is logically isolated from service S. However, the customer's device is physically surrounded by networked devices. These devices connect to one or more networks by means of different channels. Schematically, an instance of such a configuration is reported in Figure 1b.

Mobile agents. Mobile devices have heterogeneous hardware profiles, e.g., memory, computational power, presence/absence of bluetooth, etc. Also, their configuration can change over time, e.g., on device battery charge and position.

In general, we can consider each device to be a computational platform that can install and run (at least small) applications. Moreover, we assume that all the (enabled) devices run software supporting basic orchestration steps.

Communication protocols. Connected devices use different channels, e.g., bluetooth and wifi, to establish connections. These channels have different features and, in general, have been designed for different purposes. We call a *pit* a device having direct access to the internet. Hence, the devices must create a network where one or more pits are present. Other resources can be present in the network, e.g., computation and memory nodes, and they can be exploited for the service delivery.

Device contracts. Each device holds a precise description of its features and requirements, namely a *contract*. Contracts describe which kind of activities can be carried out by the device. Examples of entries of a contract are:

– Available disk space, i.e., the amount of memory that the device can offer.
– Available connections, i.e., channel types, bandwidth, etc.
– Computational capacity, i.e., whether the device can carry out computation.

Each feature can be associated to a precise cost that must be paid to access/use it. Informally, we can see a contract as a list of predicates like:

NET. Internet: 3G (Bandwidth: 3.2 MB/sec; Cost: 0.05 €/MB) + WiFi (Bandwidth: 14.4 MB/sec; Cost: 0.01 €/MB);
LINK. Bluetooth: (Bandwidth: 720 Kb/sec; Cost: 0 €/sec);
DISK. Space: 2 GB; Cost: 0.01 €/MB; Expiration time: 60';
CPU. Speed: 800 MHz; Cost: 0.02 €/sec;

For instance, the first rule says that the device can connect to the internet in two different ways (i.e., 3G and WiFi) and describes the differences in costs and bandwidth. Instead, the meaning of the third clause is that the device can offer up to 2 GB of memory space at the given cost per MB. Also, the contract says that after 60 minutes stored data will be deleted.

Other devices can retrieve a contract and compare it against their requirements. Moreover, when a contract does not satisfy certain requirements, a *negotiation* process is started. Negotiation consists in proposing modifications to the original contract clauses. If the contract owner accepts the proposed corrections, a new, suitable contract is used in place of the previous, unfitting ones.

2.2 Network orchestration

Network orchestration plays a central role. Indeed devices must cooperate in a proper way in order to achieve the network goals. Among the recent proposals for the organization of networks, *Software Defined Networking* (SDN) is receiving major attention.

Software defined networking The main feature of SDN is a clear distinction between control plane and data plane in network choreographies. Mainly this approach allows for exploiting centralised service logic for the network orchestration. Typically, network nodes take responsibility for both data transfer and network organization activities promiscuously. This behavior is acceptable when networks are composed by dedicated nodes, i.e., platforms (hardware and software) dedicated to the network management. However, under our settings we cannot expect to have homogeneity in nodes configurations. Indeed, nodes configurations may differ for many reasons, e.g., hardware, battery state, user's activities and security policies. Hence, we must expect that the network management is carried out by some dedicated devices in a partially distributed way.

Nodes offering advanced computational capabilities can take responsibility for the control activities. These activities include node orchestration, network monitoring and reaction to changes. Since nodes do not have any pre-installed orchestration software, mobile code must be generated and deployed dynamically. Figure 2 represents this process.

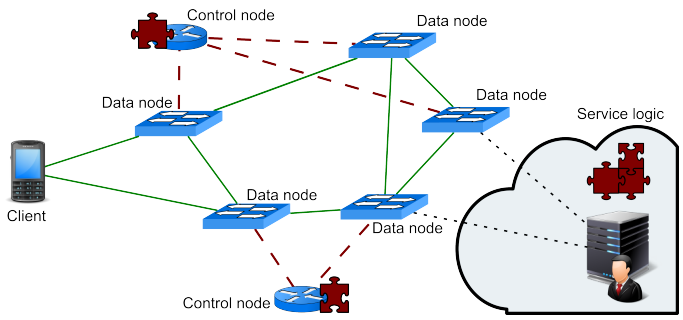


Fig. 2. Control and data nodes participating in an orchestration.

Solid lines represent data links, i.e., channels used for service-dependent traffic. Instead, dashed lines denotes control channels, i.e., used for control activities. Control instructions are generated by the service provider being the entity holding the service logic. A control node receives a piece of software (jigsaw piece) that is responsible for managing the behavior of (part of) the network.

3 Security issues

Many security threats can arise during the recruiting, negotiation and execution phases. All the security aspects can appear at either (*i*) network level or (*ii*) service level. Below, we list and describe the main security issues showing whether they affect the first or second of these layers.

3.1 Network security

Devices in opportunistic networks build transient and goal-driven networks, thus behaving as peers. Hence, most of the security issues at network level resemble those of P2P networks. By joining an opportunistic network, each device gets potentially unknown neighbours and exchange data with them. In this context, confidentiality and integrity are major guarantees to provide to the final user, since information sent to the service S may be corrupted or intercepted by malicious devices. Device authentication is also required in order to recognize and isolate malicious devices.

Authenticity. Usually devices are uniquely characterized, e.g., by the MAC address or IMEI code. However, a strong authentication relating a device with a physical user is hard to achieve at this level. Also a global authentication in a network is hardly achievable, due to the lack of a central authority and the heterogeneity of device platforms. However, mutual and pairwise authentication between devices may be easily carried out. In this context, from the single-device point of view the authentication is aimed at 1) allowing honest devices to

recognize and isolate malicious ones, and 2) building temporary trust relationship between trusted and authenticated devices in order to share bandwidth, memory and disk resources. To meet these targets, the adoption of gossiping algorithms [6], combined with cooperative access control mechanisms [12] can be adopted.

Confidentiality. Message confidentiality is a main concern since a device reaches the service by sending data through unknown and untrusted devices, without the possibility to trace its own traffic. However, confidentiality at this layer can be granted by the use of secure channels built at higher layer. For instance, HTTPS channels established between the source device and the service are suffice to provide the required confidentiality through traffic encryption. Secrecy provided by HTTPS channels is not easily breakable, in particular for single devices in the networks.

Integrity. Ciphering data grants secrecy but does not prevent devices from tampering the traffic they receive. Thus, the use of integrity scheme can be envisaged in opportunistic networks. There exist integrity schemes based on shared keys and private/public key. The choice between shared key (e.g., MAC schemes [14]) and public/private key schemes (e.g., DS schemes [4] and Batch verification schemes [10]) depends on the contingency of the opportunistic network. Besides, the use of such schemes requires, at most, the installation of simple libraries or programs on the device.

3.2 Service level security

Here we can identify two groups of entities aiming at different security goals: (*i*) the service-customer coalitions and (*ii*) the control-data nodes.

Service-customer security. The service provider and its customers share a common goal, i.e., enabling the customer to access the service according to a given SLA. Among the clauses of the agreement, security policies prescribes how the service handles the customer's data and resources. In general, the provider can rely on a trusted service infrastructure. However, in our scenario the service is delivered by a group of, potentially untrusted, devices which extend the trusted infrastructure. Intruders could join the network and perform some disrupting attack, e.g., denial of service, also depending on the service typology.

On the other hand, the service can include security instructions in the code deployed on control nodes. In this way, control nodes can monitor the behavior of (a portion of) the network. Monitoring allows control nodes to detect intruders and, possibly, cut them off. Even more dangerously, the intruder could be a control node. In this case, the service can detect the misbehaving control node by directly monitoring its behavior. Control node monitoring can rely on other control nodes, including the (trusted) customer. Hence, a group of control nodes can isolate a malicious peer when detected. Still, control nodes collusion represent a major threat and mitigation techniques could be in order.

Nodes security. Data and control nodes have a different security goal. Since they join an orchestration upon contract acceptance, their main concern is about avoiding contract violations. Being only responsible for packets transmission, data nodes can directly enforce their contract via resources usage constraints.

Control nodes require more attention. As a matter of fact, they receive orchestration code from the server and execute it. The execution of malicious software can lead to disruptive access and usage of the resources of the device. Thus, a control node must have strong, possibly formal, guarantees that the mobile code is harmless, i.e., it respects the given contract.

A possible solution consists in running the received code together with a security monitor. Security monitors follow the execution of a program and, when they observe an illegal behavior, run a reaction procedure, e.g., they can throw a security exception. A security monitor comparing the mobile code execution against a given contract is an effective way to ensure that no violations take place. Although monitoring requires extra computational effort, lightweight implementations causing small overheads have been proposed, e.g., see [8].

Another approach exploits static verification on to avoid run-time checks. Briefly, the service provider can run formal verification tools, e.g., a model checker, before delivering the mobile code. The model checker verifies whether the code satisfies a given specification, i.e., the contract, and, if it is not the case, returns an example of a contract violation. Instead, if the process succeeds, a proof of compliance is generated. Using *proof-carrying code* [13] the proof is then attached to the code and, then, efficiently verified by the control node. A valid proof ensure that the code execution cannot violate the contract of the node.

4 Meeting at the airport: a case study

We consider the following scenario. A e-meeting service offers to its customers the possibility to organise and attend to virtual meetings. A meeting consists of a group of customers that use (i) a VoIP system for many-to-many conversations and (ii) file sharing for concurrently reading and writing documents.

Private companies buy annual licenses. Then, employees install a free client application on their devices and access the service using proper, company-provided credentials. Nevertheless, company employees use to travel frequently and, often, need to buy wireless access in airports and train stations. This practice causes extra, variable charges on the service usage.

Service requirements The two service components, i.e., VoIP and file sharing, have different features. Mainly, the VoIP service has precise constraints on transmission times in order to make the communication effective. In order to respect this constraint, the service can reduce the voice encoding (up to a minimal threshold) quality whenever slow connections risk to cause delays.

Instead, the file sharing system must guarantee that documents are managed properly. Roughly, users can acquire the ownership of a document and modify it until they decide to release the control. Each time a document is retrieved

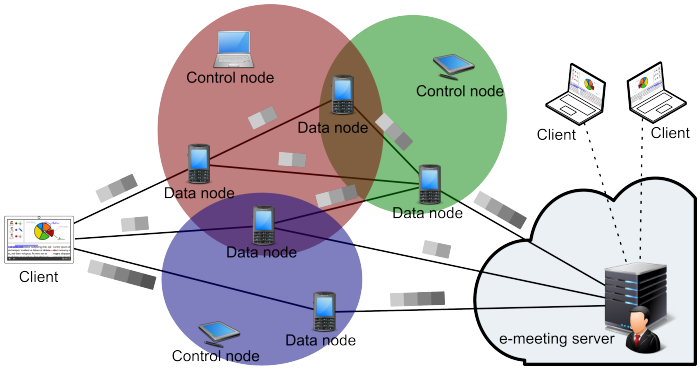


Fig. 3. Orchestration providing opportunistic access to e-meeting.

(submitted) it is downloaded from (uploaded to) a network repository database. Document loading and saving are not time critical operations, i.e., they can be delayed, but data consistency must be guaranteed.

Network structure In order to set up a suitable network, the client starts a *recruiting* procedure. Briefly, it floods with a request message its neighbours (up to a fixed hops number) and collects their contracts. If the set of received contracts satisfies preliminary conditions, e.g., sufficient nodes density ad existence of internet-enabled nodes, the negotiation process starts.

Negotiation requires interaction with the web service. To do this, at least one of the nodes having internet access must take responsibility for sending the negotiation information to the orchestration service. This information includes nodes contracts and topology description, i.e., nodes neighbours tables. The orchestrator check whether the network configuration satisfies minimal requirements and returns contract proposals for the control nodes. The nodes receive the negotiated contracts and decide whether to accept or reject it. If a contract is rejected, the process can be repeated[6]. When all the control nodes accept the proposed contracts the service send them a piece of software implementing part of the distributed orchestration algorithm. Each node verifies the validity of the received code and starts the orchestration procedure. The resulting network organization is depicted in the figure below.

Intuitively, each control node is responsible for coordinating the activities of a group of data nodes (rounded areas). Data nodes are responsible for transmitting network traffic and they are recruited and managed by control nodes. Also, control nodes must react to a plethora of possible events, e.g., topology changes, data and control node fall or performances decay.

[6] We assume that nodes cannot reject a contract respecting its own original clauses. For instance a node offering 2 GB of disk space can reject a request for 3 GB but not one for 1 GB.

5 Related Work

Many technologies are related to our model. Here we briefly describe those that, in our view, better apply to this context.

Just recently, software defined networking received major attention. Among the others, OpenFlow [11, 9, 15] is the leading proposal of an implementation of SDN. Basically, OpenFlow allows network managers to programmatically modify the behavior of networks. Although it is currently applied to standard network infrastructure, i.e., standard routers and switches, this technology seems to be also suitable for mobile devices. Hence, we consider it to be a promising candidate for the implementation of orchestration tools.

Formal verification plays a central role under our assumptions and it appears at several stages. Mainly, contract-based agreements require formal analysis techniques for granting that implementations satisfy a contract. A standard method for this is model checking [7, 2]. However, also proof verification is crucial for allowing network nodes to check the proof validity when the source in an untrusted service. This step can be implemented by using proof-carrying code [13].

Being a main concern, code mobility and composition environment must include proper security support. In particular, policies composition techniques must be included in any proposed framework. Local policies [3] represent a viable direction for allowing several actors to define their own security policies, apply them to a local scope and compose global security rules efficiently. Also, since our proposal is based on mobile devices technology, specific security solutions for mobile OSes must be considered. In this sense, in [1] the problem of securing the Android platform against malicious applications has been studied.

Finally, also dynamic monitoring appear to be necessary for managing and re-organizing the network in case of necessity, e.g., upon failure discovery. A possible solution consists in using the approach presented by Bertolino et al. [5] for retrieving and collecting information about nodes behavior. Instead, for what concerns security monitoring, a possible approach is presented in [8]. Since this proposal has been tested on resource limited devices, it seems a good candidate for avoiding computational loads on network nodes.

6 Conclusion

In this paper, we described the possibility of applying Software Defined Networking (SDN) paradigm as potential solution to build and manage opportunistic connection among mobile devices and web services. In particular, we described the main issues arising when trying to orchestrate devices that share the goal of implementing a QoS compliant network. Also, we considered the security issues deriving from such a model and possible approaches and countermeasures. Finally, we presented a case study that highlights the main aspects that must be considered under our assumptions.

References

1. A. Armando, G. Costa, and A. Merlo. Formal modeling and reasoning about the Android security framework. In *Proc. of 7th International Symposium on Trustworthy Global Computing*, 2012. (To appear).
2. C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
3. M. Bartoletti, P. Degano, G. Ferrari, and R. Zunino. Local policies for resource usage analysis. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(6):23, 2009.
4. M. Bellare and P. Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In *Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'96, pages 399–416, Berlin, Heidelberg, 1996. Springer-Verlag.
5. A. Bertolino, A. Calabrò, F. Lonetti, A. D. Marco, and A. Sabetta. Towards a model-driven infrastructure for runtime monitoring. In *Proceedings of the Third International Workshop on Software Engineering for Resilient Systems*, pages 130–144, 2011.
6. S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip algorithms: Design, analysis and applications. In *in Proceedings of IEEE INFOCOM*, pages 1653–1664, 2005.
7. E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.
8. G. Costa, F. Martinelli, P. Mori, C. Schaefer, and T. Walter. Runtime monitoring for next generation java me platform. *Computers & Security*, 29(1):74–87, 2010.
9. B. P. et al. *OpenFlow Switch Specification*. OpenFlow, feb 2011. Available at http://www.openflow.org/documents/openflow-spec-v1.1.0.pdf.
10. P. Gasti, A. Merlo, G. Ciaccio, and G. Chiola. On the integrity of network coding-based anonymous p2p file sharing networks. In *Proceedings of the 2010 Ninth IEEE International Symposium on Network Computing and Applications*, NCA '10, pages 192–197, Washington, DC, USA, 2010. IEEE Computer Society.
11. N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, Mar. 2008.
12. A. Merlo. Secure cooperative access control on grid. *Future Generation Computer Systems*, 29(2):497 – 508, 2013.
13. G. C. Necula. Proof-carrying code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '97, pages 106–119, New York, NY, USA, 1997. ACM.
14. B. Preneel and P. C. Van Oorschot. On the security of two mac algorithms. In *Proceedings of the 15th annual international conference on Theory and application of cryptographic techniques*, EUROCRYPT'96, pages 19–32, Berlin, Heidelberg, 1996. Springer-Verlag.
15. A. Tootoonchian and Y. Ganjali. Hyperflow: a distributed control plane for openflow. In *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pages 3–9, Berkeley, CA, USA, 2010. USENIX Association.