Context Sensitive Smart Device Command Recognition and Negotiation

Frank Bauerle, Grant Miller(✉), Nader Nassar, Tamer Nassar, and Irene Penney

IBM Corp., Somers, NY, USA
{fbauerle,millerg,nnassar,tamer,sprouts}@us.ibm.com

Abstract. The Internet of Things (IoT) represents a dramatic shift in how computing and communication interrelate. Devices can potentially connect to anything at any time in any place. While many researchers focus on how to make the IoT seamless, faster, and add more devices into the mesh, our approach is to introduce a new approach whereby devices not only communicate, but also have a level of context awareness that results in optimal intelligence among the mesh of connected devices.

Keywords: Iot · Negotiation · Context-sensitve · Mobile

1 Introduction

The numbers of smart devices are increasing almost exponentially. The capabilities of these devices are increasing daily. The ability for these devices to interact is increasing exponentially as mobile and telecom networks continue to expand. In the past, smart devices and their associated network were built for purpose. A network of medical devices knows which other devices are or may be available and understand how to interact.

It is when an unknown number or type of devices exist on the same network that the specifics of how these devices interact and respond to commands becomes a problem. An algorithm and approach for determining which device is best suited to respond to a command must be implemented.

1.1 Background

Imagine walking into a room full of smart devices. There are tablets and phones. An Xbox360 is sitting next to the SmartTV. A DVR sits next to the television. Smart remotes sit on the coffee table. A laptop sits on the coffee table (Fig. 1).

When a command[1] is made to turn the television on, which device should respond? When a command is issued to watch a movie, which device should respond?

[1] Throughout this paper, unless the situation is specific to voice commands, we use the term "command" to refer to directives that are given by any input method (voice, keyboard, or other).

R. Giaffreda et al. (Eds.): IoT360 2014, Part I, LNICST 150, pp. 314–330, 2015.
DOI: 10.1007/978-3-319-19656-5_44

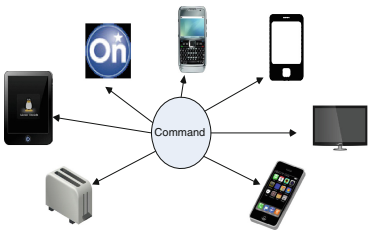

Fig. 1. Command to smart devices

Today, it seems simple. We know which device the movie is on or which remote we need to use to turn on the television.

In the future, this will not always be the case. There will be multiple smart devices that can respond to a command whether it is a smart phone or tablet that have embedded television remote applications or the Xbox360® that can respond to a command to play a movie that may also be available on the television or DVR.

As smart devices become smarter, the ability for these different devices to recognize and respond to different commands will only increase. If multiple smart devices attempt to respond to a command, the result could be chaos. A movie could begin playing on the Xbox, the television, and even the Netflix™ application on a tablet.

To help bring order to this potential chaos of having a multitude of devices respond to a command, a method of having these smart devices negotiate amongst themselves to determine which device should respond is needed.

In order to support this capability, it is clear that smart devices will need to evolve. Given the rate at which these devices are currently evolving, it is not a stretch to assume that in the near future smart devices will become smarter and have the ability determine through the context of the device, its location, and the type of command issued, which is the best device to respond.

In this paper, we lay out the case for a context-aware negotiation-based command determination method.

1.2 The Evolution of Smart Devices

There are many types of devices becoming connected. There are many examples of "smart things" that exist, such as wireless sensor and actuator networks, and embedded devices [1]. Smart devices began as single purpose devices. They were purpose-built to satisfy a single use, or command, or set of commands that they were built to satisfy. A thermostat is a great example of a built-for-purpose smart device. An individual can interact with the device to set the temperature in the house. The same individual can create a program to raise and lower the thermostat during the course of a day. But, other than potentially providing some history on the thermostat settings that is all the thermostat can do.

These built-for-purpose smart devices are typically hardware based and have limited ability to learn.

Today, multipurpose devices are becoming more prevalent. These are devices that are capable of fulfilling a range of requests. A smart phone or next-generation television are excellent examples of multi-purpose devices. They are capable of responding to multiple types of requests. They can access the Internet. They can play movies. They can make phone calls.

Multi-purpose devices can sometimes have also primary and secondary purposes – the primary purpose of a television is obvious. As part of our interconnected world, televisions are now able to connect to the Internet. Because of these secondary capabilities, devices such as smart televisions may be able to fulfill multiple types of request – but may not be the most appropriate device in the network to respond to a request.

1.3 Problems with Current Networks of Interconnected Devices

When the network of devices is known and the purpose and function of each device is clearly defined, it is relatively straight forward and simple to define which device should respond to which command.

It is when there are several, or many independent devices available for a similar task that challenges rise. Part of the challenge is to achieve interoperability between connected devices while maintaining their "smartness" and allowing for adaptive and autonomous behavior [2]. This implies that the devices need to remain true to their purpose, but also capable of negotiating with other devices trying to fulfill similar tasks.

There are precursors to the connectivity of autonomous devices to build from. Sensor networks are collections of connected devices that are autonomous, but coordinate and communicate for a single purpose. Each device in the network has the same task and is responsible for collecting and transmitting data. The design of these networks focused on coordination to transmit data (see Fig. 2).

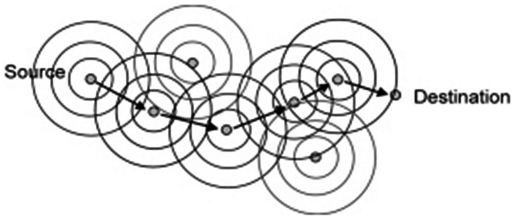

Fig. 2. Source-destination

Early challenges with sensor networks, such as network discovery, control and routing, collaborative information processing, tasking and querying, and security all set a foundation for connecting smart devices and the basis for this proposal [3]. Pervasive computing also set a foundation for communication between devices and coordination around responding to requests [4].

Networks of smart devices have advanced to become ecosystems of devices all collectively providing a function for a "user". Examples of hospitals with many monitoring devices or automated homes such as shown in Fig. 3 with devices provided an array of functions.

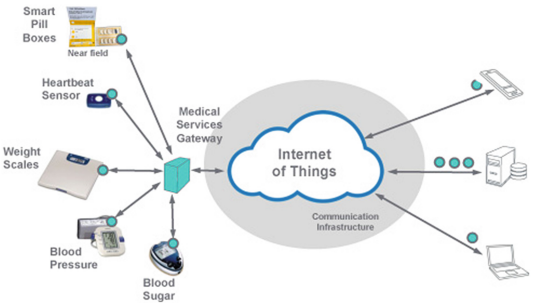

Fig. 3. Internet of Things

Even in these advanced ecosystems, the coordination and negotiation between devices is not for who can fulfill a request, but connecting, authorizing, and transferring collected data.

When several devices can fulfil a request and only one is required to do so, a whole new type of negotiation is needed. The next figure represents the high level steps required for this next evolution of negotiation (Fig. 4).

Fig. 4. Negotiation as a service process overview

The first step is discovery and identification: what the request being made, who is requesting it, and why it is needed. This request might be sent to multiple devices all capable of responding to a request. Once discovered and identified, the devices need to collaborate on a proper response. Previous work has been done providing a coordinated layer response [5].

The proposal for this paper is to go beyond the existing work. Many proposals focus on a centralized coordinator for devices, which may be possible in dedicated networks of known systems [1]. In an IoT environment, each device is autonomous and may not exist in a known or dedicated environment. To do that, the context of a request needs to be taken into account. A coordinated response among entities using services, nodes, equipment, and the infrastructure must use a specific context [6].

2 Proposed Solution

When the network and paths are not predefined, the ability of this dynamic network of smart devices to interact with each other becomes much more difficult. Our goal is to establish negotiation between a set of existing intelligent devices, all of which are capable of performing the same or similar task, to determine which device should perform a task.

In order to determine whether to respond to a command, the devices need to be aware of a number of variables. They need to be aware of where they are. They need to be aware of who issued the command. They need to ultimately determine through negotiation which device is best suited to respond.

2.1 Solution Overview

In this paper, we describe a solution and approach by which a dynamic network of smart devices can negotiate among themselves to determine which device is algorithmically better to respond to a given command.

Our focus is on two aspects of this solution.

1. The devices themselves – what must they be capable of supporting and doing to support the contextual response required. The devices must be capable of responding to a given command in order to be viable.
2. The interactions between the devices - how must the interactions be identified, the context understood, and the negotiations between the devices be done to determine best fit and most appropriate response

It must be clearly stated that being context aware is key to this solution. The devices and the device interactions must all be aware of the context of the command.

Within this solution, a command is given – spoken, sent, etc. – over standard protocols – and the appropriate devices respond to the request.

The response must be given in context – meaning that the responding device must understand:

(1) Is it capable of responding – can it fulfill the command being given?
(2) Where is the device now – understanding the context of location and network
(3) Who gave the command – understanding the context of the command giver
(4) Is the device authorized to respond to the command?
(5) Is it the appropriate device to respond? Is there another device in the network that can fulfill the response more effectively?

In this context sensitive device interaction, when a command is spoken, there may be multiple devices that are capable of responding. The following diagram shows a command being issued to a potential set of devices (Fig. 5).

These devices must understand that there are other devices in the area that may be capable of responding. When a command is given, the devices must collaborate, and based on the available contexts (who gave the command, where was the command given, etc.), determine how to respond. As the following diagram shows, as the number of enabled smart devices grows, the networks of devices can become increasingly complex. Having a standard way of determining which device should respond will become increasingly important (Fig. 6).

The first decision is both the simplest and potentially the most complex to address. For single-purpose devices, the decision is simple – am I equipped to fulfill the request or am I not. A toaster is not capable of responding to a request to turn on the television – or to find directions to the nearest sushi restaurant.

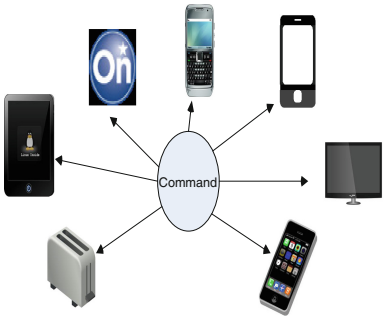

Fig. 5. Command issued

Fig. 6. Devices interact

The simplest case is when there is a single device in the area that can fulfill a request. For example, if a request is made to toast some bread, the only device that has the capability to perform that action is the toaster (Fig. 7).

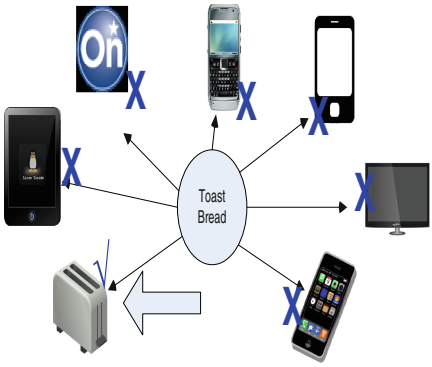

Fig. 7. Device wins simple negotiation

In this instance, the toaster would attempt to negotiate with any devices that might be able to fulfill the request. Finding no other devices in the vicinity that can fulfill the request, the toaster would then perform the action.

Using a more complex example, if a command is given, such as 'turn on the television', there may be multiple devices that can fulfill the request. Smart phones and tablets now can run applications that allow the device to perform as a television remote control. The second question must be addressed by the network of devices – which device should respond (Fig. 8)?

Fig. 8. Multiple devices are capable of responding

In this example, when the command is given, the devices must first answer the question "can I fulfill the request?" For the purpose of simplicity, only 2 of the smart phones, the tablet, and the smart television are all capable of turning on the television. The question that must be answered is – which device (of those four) is the most appropriate to respond to the request.

In the example above, the television itself is the most appropriate device to respond and so, responds by turning on the television (Fig. 9).

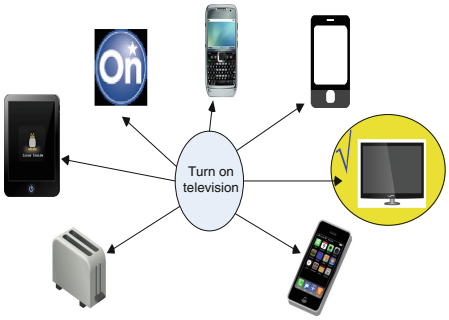

Fig. 9. Devices wins complex negotiation

Another factor is understanding and determining which devices are authorized to respond. A device may be capable of responding – but it may not be authorized to respond.

In defining this, we look at both implicit and explicit authorization, as well as rule sets. Implicit authorization may be leveraged for simple commands – which device should turn on the television. When the devices negotiate, there may be multiple devices that are capable of responding. The devices may have no explicit rules set up for responding (or not responding). Instead, the device that best satisfies the rules will win the handshake between the devices.

In the example of a command being given to 'turn on the television', the network of devices must determine which devices can respond. During negotiations, those devices that are capable of responding should determine which device is implicitly the best device to respond – in this instance, the television is capable of turning on directly without going through a remote. In this instance, the television will turn on directly.

Explicit authorization exists in this model to satisfy multiple situations. First, the owner of the device may elect to change the rules so that the remote takes precedence over the smart television. In this situation, when the command is given, the rule establishing the order of precedence and which device should respond has been set to authorize the smart remote instead of the smart television directly.

In other situations, explicit authorization is important to prevent accidental or intentional misuse. If a command to 'phone home' is given, the devices must be intelligent enough to understand the nature of the request and who is typically associated with that request. In a network of devices, each device must understand that based on the context of the request that it and it alone are qualified to make the request.

Before responding to the request, the device should know who gave the request. It may not be sufficient for the device to simply respond to the request based on whether it was capable of responding or was the best device based on feature/function. It must also be authorized to respond. In this case, commands to dial the phone is restricted to the phones owner (Fig. 10).

If the person who spoke the command is not the phones owner, the request will be ignored. Another smart device in the network may be set up to respond to that request – but the smart device that we are referring to is explicitly set up to only respond to the owners voice for this particular command.

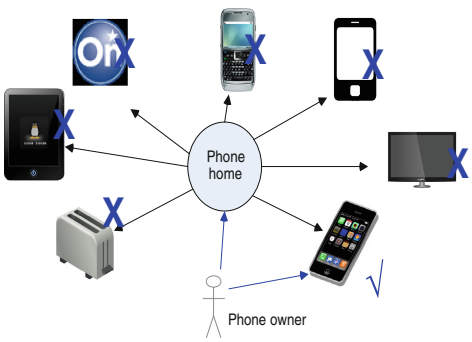

Fig. 10. Context sensitive – device owner

Finally, the device must be aware of where it is located. Many devices today have this capability in some limited fashion. For example, smart phones and tablets are aware of surrounding wi-fi networks – and automatically connect to the appropriate SSID when the opportunity presents itself – as well as being aware of their physical location via GPS coordinates.

This solution carries the analogy a bit further. The smart device must know where it is located in order to know whether it should respond to a particular request (Fig. 11).

Fig. 11. Context sensitive – device location

In the case when a call to 911 is made in a vehicle, the enabled smart devices must recognize where they are. As the devices begin negotiating to determine which device should respond, the Onstar device should assert that it is the most appropriate device to respond because it is part of the vehicle – as such, it should take responsibility for making the call. Based on the negotiation rules, all other enabled smart devices should submit to this assertion.

2.2 The Solution Stack

Logically, the solution adds a 'negotiation layer' to smart devices that helps the devices negotiate and determine which device is the best device to respond (Fig. 12).

The solution stack diagram above describes the basic device stack. It begins once a command is given – either to a software-enabled smart device or a hardware-enabled smart device. These commands are received and responses provided on a defined transport layer.

Within the device, once the command is received, the device must determine whether it is capable of responding to a given command. A decision not to respond can be made because the device is not capable of responding or because according to the rules associated with the command on the device, it is not authorized to respond.

Starting from the top of the stack and moving down,

1. The transport and protocol for communicating between devices should not matter to this solution. Because Bluetooth® technology is prevalent today, we assume that it

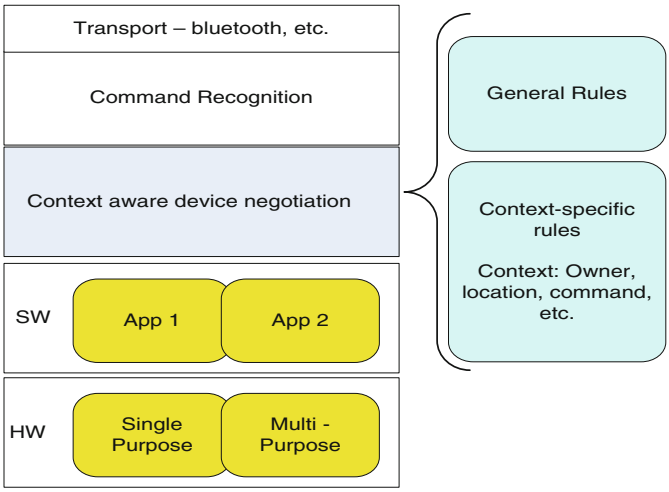

Fig. 12. Smart device solution stack

is the transport for the inter-device communications today, but it is not a requirement for our proposal. As technologies evolve, other transport technologies may take precedence and become the standard.

2. All devices must have the capability of recognizing the command - and be able to respond based on some set of rules. This context-aware negotiation module is used to determine whether a device should respond to a given command and to negotiate with other devices on which device in the network is most appropriate to respond.
3. Assuming more complex smart devices, there are special purpose software modules that may exist on the device and that may be able to respond to a request.
4. All devices will have some level of hardware involved - this layer is shown to capture the fact that both simple and complex devices are part of this ecosystem.

Responding to a Command. Assuming a device is capable and authorized to respond, when a command is recognized, the context aware device negotiation module is invoked and needs to determine how to respond.

When a command is issued, the sequence for determining which device responds should be as shown in Fig. 13 - Command Processing.

Defining the Rules for Responding to a Command. When the command is received by a device, the device should determine whether it is capable of fulfilling the request. Each of the devices that validate, authenticate, and authorize the command, will then negotiate with the other devices in the network to determine which device is best suited to respond.

There will be times when the negotiation is simple – in the case of a smart toaster in a network of smart devices, only the toaster is capable of responding to a command to 'toast bread'.

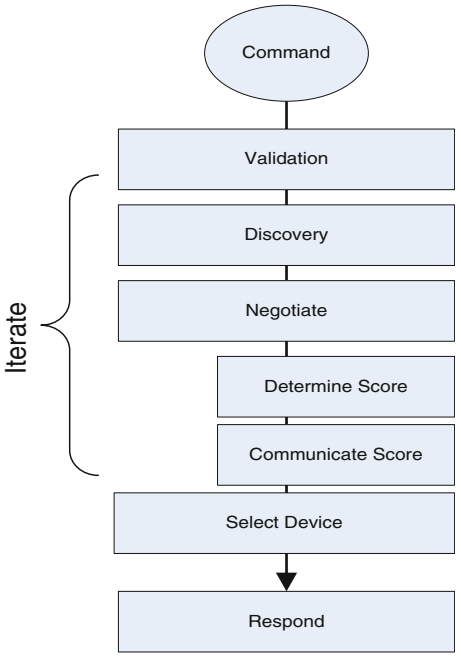

Fig. 13. command processing

In other instances, multiple devices may be capable of responding to a command. In this instance, a set of rules to sort and prioritize devices must be established. Rules should be established to understand

- Who spoke the command? Is the owner of the device the person who spoke the command?
- Where was the command spoken?
  - Was it spoken in a car?
  - What network is the device on currently? Wi-fi? 4G? etc.
- What is the social weighting of the application?
- How did the owner of the device rate or prioritize the applications on their device?

Weights or scores will be assigned to the different contexts and rules to help the devices respond with their ability to respond and their worthiness of responding. Weight may be given to the owner of the device in many cases to allow device affinity. In other cases, the network or connectivity of a device to a network – whether the device is LTE or 4G capable and can respond on a higher speed network - may skew the command toward a particular device.

Using the iteration or recursive model, the devices will broadcast their scores to the other devices. The device with the highest score will be confirmed – that device will then respond to the command.

Table 1. Proposed rules

Rule	Description
First in, first out	First device to respond to the command takes precedence
Owner of device	if voice is recognized as owner, takes priority. If the command is given using the owners voice, that device should take precedence.
Device purpose	If the device has a specialized purpose, it should (or could) take precedence.
Command location	Where was the command issued? Was it on a local network? Was it on a wi-fi network? Was it on the devices 'home network'?
Device connectivity or utilization	A rule could be set up to determine if connectivity or responding to a command would cost the owner. example - wireless vs. wifi, exceeding monthly limits
Quality of signal	For some commands, quality of signal may be critical. A device that has a higher quality of signal might need to take precedence.
Timestamp comparison	earliest timestamp takes precedence examine correlated packets - use packet timestamp to concede to packet with earliest timestamp
Deadlock break	Multiple devices may receive and indicate their ability to respond almost simultaneously. A method of breaking deadlocks must exist
<other rules>	…

If it is determined that the message is a new command, a set of rules will be evaluated - the attached table provides an example set of potential rules that could be embedded within each device (Table 1).

For some devices, the rules will be rigid - especially for single purpose devices. The rules for those devices that are multi-purpose may be customizable - through the device settings - or through a special application.

Where the time to respond to a command is critical, rules may be established to immediately execute the command without iteratively negotiating with other devices. In the case of a call to emergency services, it is more critical to reach emergency services than to determine the best device to execute the call. To prevent mis-use of this feature, rules may be embedded to only allow commands to be completed if initiated by the device owner.

Rules can be stored locally or centrally – depending on how the solution is ultimately implemented, there can be multiple layers of rules – general and context specific.

- Static rules will exist when a device only has one purpose (in the case of a toaster who can only toast bread) or soft rules
- Context sensitive rules will exist when the context of a command needs to be evaluated before responding to a request.

Whether stored centrally or distributed on each device, the rules must all be uniquely identified and associated with a given command. When the device recognizes the

Table 2. Define rules

Attribute	Definition
Command	Command that rule applies to
Rule ID	Rule identifier
Rule	Rule definition
Priority	User or system defined priority of the command
Weight	Weight of rule
Time window	Time window in which rule applies

command, it would retrieve or consult the rules for that command and determine whether the device should respond and if so, how it should respond (Table 2).

Negotiating Iterating to Determine Which Device Should Respond. Once a device receives a command and responds with its intent to respond, it should evaluate the rules for the given command and respond to the network of other devices that have indicated they can satisfy the command.

The device will evaluate the rules and respond with a message that indicates the command that was given, the timestamp that the device responded, the device-specific command ID, and the other information (Table 3).

Table 3. Response message format

Attribute	Definition
Command	Command given
Timestamp	Timestamp the device responded to command
Command ID	Unique identifier for the command on the device
Score	Score from evaluating the rules for the context-aware device

After evaluating the rules, each receiving device will respond with a status - it could indicate that it is a viable device - it could also concede because it doesn't fit the rules.

There may be instances when the weights or scores reported by the devices in a network will be the same – in that case, a set of rules to break the deadlock condition should be established – in many cases, the first device to react to the negotiation request will be given priority to respond to the command.

In the example of a command to 'find a sushi restaurant', each of the devices capable of providing a response to that command should consult their rules database and determine the weight or score of their response.

The score may be influenced by the default application on a particular device, who spoke the command, or the location of the device. In the example of a device owner asking, the device may assign a higher score because of device ownership – or even the fact that the device is OnStar capable device and the command was spoken in a car.

The device will communicate their score to the other devices in the network. Those that are still vying to fulfill the request will evaluate the scores against their score.

The devices will broadcast a response indicating whether they want to assume control or will submit to one of the other devices in the network.

Using the context sensitive device negotiation solution, the devices would interact with each other and based on a set of general or context-specific rules determine how to respond.

The devices will iterate in the context sensitive negotiation phase until only a single device responds. With our solution, devices can understand – based on the network of other devices and the rules governing the device – which device should respond to a request. With the ever increasing proliferation of smart devices in our world, being able to issue a command and have the appropriate device respond is becoming increasingly important. In this solution, based on a command, the enabled devices in the network can determine which devices are capable of responding and then negotiate which device should respond to the request.

2.3 Negotiating as a Service

The negotiation described thus far is peer-to-peer between devices to determine the best device between a set of acknowledged devices. In some situations, this may not be enough.

There are cases where 2 or more devices may be capable of handling a request just as well as any other. In these cases, an arbitrator could help decide which device to select.

Additionally, in a known location, you may want to control which device responds to a command even if you don't "own" the devices in the network. Examples of this include a hospital device ecosystem where you only want certain devices responding and you want to control which devices respond in a given situation. Even though a device may be authorized to be on the network, you may not want it to perform some actions.

Another example can be found in the workplace where smart phones are often allowed to access the corporate network. You may want to authorize the phone to respond to some commands but not others – for example, "download file" may not be a command that you want to allow your phone to fulfill.

Negotiation as a service is centered by the existence of an arbitrator. The arbitrator would understand which devices, commands, and rules it may arbitrate. The arbitrator would also recognize all commands and receive all responses from available devices. The arbitrator would then arbitrate and evaluate all responses – and determine which device should respond to a given command. The next figure shows a high level view of this (Fig. 14).

The negotiation starts with each device that has similar capabilities contacting the arbitrator and sending its identification. The arbitrator uses the identity of the devices and their capabilities to determine the appropriate device to respond.

The next 2 figures illustrate this interaction (Fig. 15).

The processing done by the arbitrator includes a similar set of rules that were described in the peer-to-peer model. After the arbitrator selects a device, the user may

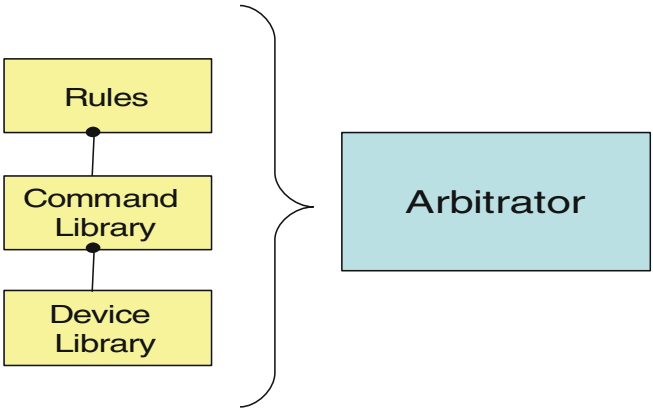

Fig. 14. Rules arbitration

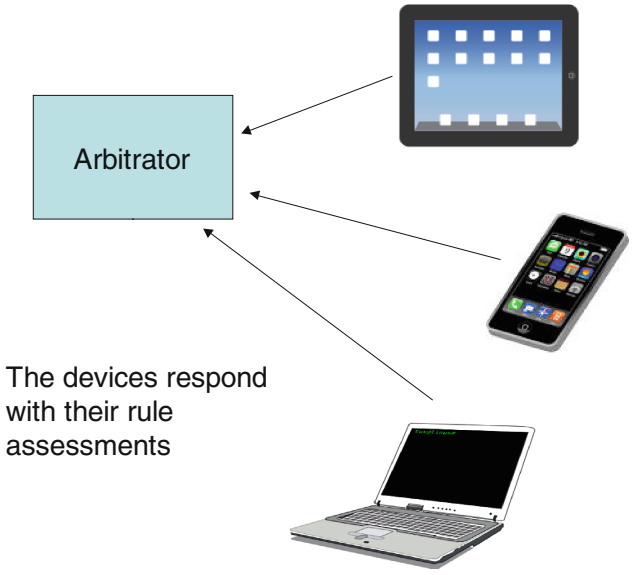

Fig. 15. Arbitrator asssess response

override that selection for another device. That override is collected and saved as a rule adjustment by the arbitrator. In this way, a social aspect of arbitration is introduced as people build up the rule engine for which devices work in certain conditions for certain sets of devices (Fig. 16).

Arbitrators can exist in multiple configurations. They can be distributed, centralized, or tiered. Tiered arbitrators can first department, then division, then corporate-wide.

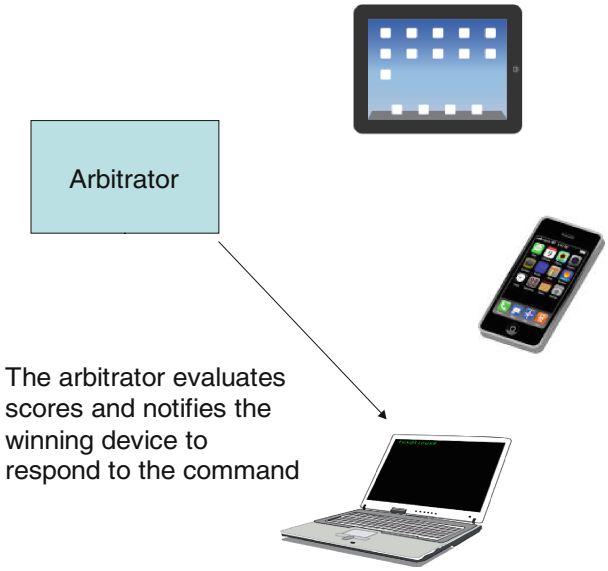

Fig. 16. Arbitrator selects device

3 Conclusion

In conclusion, we introduced a new approach focusing on context awareness that could be implemented in the Internet of Things. Commands are responded to based on many factors that theIoT could negotiate to determine the best device to respond. This approach would add an additional layer of intelligence into the IoT landscape. This approach paves the way for IoTaaS where IoT are integrated through a layer of context awareness that enables the whole paradigm as a service.

We also introduced how negotiation could be done as a service to give and addition layer of "intelligence" to the IoT device ecosystem.

4 Future Work

There are several areas we have identified that warrant further investigation. One of these areas is around security. If you, or someone else, gives a verbal command that a set of IoT devices are capable of responding to, but not owned by the requestor, how is that handled? Can I request that my neighbors' oven turn on? A level of authentication and authorization needs to be introduced into the system.

We are planning to build a prototype to illustrate the context awareness aspect and command negotiation in IoT. In addition, we are looking at the concept of device training where the IoT is updated or reduced by one or more devices. How the device introduce itself to the mesh of things, and how the network will be updated to included new contexts that been monitored by the new devices.

References

1. Guinard, D., Trifa, V., Wilde, E.: A resource oriented architecture for the Web of Things. In: Internet of Things (IOT), pp 1–8, December 2010
2. Heuser, L., Nochta, Z., Trunk, N.-C.: ICT Shaping the World: A Scientific View. Wiley Publication, ETSI, London (2008)
3. Chong, C.-Y., Kumar, S.P.: Sensor networks: evolution, opportunities, and challenges. Proc. IEEE **91**(8), 1247–1256 (2003)
4. Saha, D., Mukherjee, A.: Pervasive computing: a paradigm for the 21st century. Computer **36** (3), 25–31 (2003)
5. Future internet: The Internet of Things. In: 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. V5-376–V5-380, 20–22 Augest 2010
6. Sarma, A.C., Girão, J.: Identities in the Future Internet of Things. Wirel. Pers. Commun. **49** (3), 353–363 (2009). Find out how to access preview-only content