Pervasive and Mobile Computing 27 (2016) 133–154

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Exploring conflicts in rule-based sensor networks

Evan Magill [a,*], Jesse Blum [b]

[a] Computing Science and Mathematics, University of Stirling, Stirling, FK9 4LA, UK
[b] Horizon Digital Economy Research, University of Nottingham, Nottingham, NG7 2TU, UK

ARTICLE INFO

Article history:
Received 25 September 2013
Received in revised form 4 November 2014
Accepted 13 August 2015
Available online 31 August 2015

Keywords:
Rule-based systems
Run-time programming
Feature interactions
Rule conflict

ABSTRACT

This paper addresses rule conflicts within wireless sensor networks. The work is situated within psychiatric ambulatory assessment settings where patients are monitored in and around their homes. Detecting behaviours within these settings favours sensor networks, while scalability and resource concerns favour processing data on smart nodes incorporating rule engines. Such monitoring involves personalisation, thereby becoming important to program node rules *on the fly*. Since rules may originate from distinct sources and change over time, methods are required to maintain rule consistency. Drawing on lessons from Feature Interaction, the paper contributes novel approaches for detecting and resolving rule-conflict across sensor networks.

1. Introduction

This paper addresses rule conflict within sensor networks. There is an increasing interest in changing the behaviour of sensor networks once they are *in situ*. This is of importance in healthcare applications such as Ambulatory Assessment where networks need to be tuned to individuals. One means of providing this capability is through the use of rules; where network nodes each use rule engines to process rules distributed across the network. The rules dictate the behaviour of each node in response to sensor data values, and so control the overall behaviour of the sensor network.

As the networks are in effect programmable, it is important (as with any means of programming) that the rules act coherently to provide the desired behaviour. However if care is not taken some rules may produce conflicting behaviour and so it is imperative to detect and resolve these conflicts. Rule conflict detection and resolution must be embedded within such systems as a particular network may have a unique configuration of rules. In addition the rules may originate from different sources as settings such as healthcare have a range of stakeholders. Also as some systems allow rules to be distributed on the fly on a real-time basis, detection and resolution techniques must be embedded to respond to changing circumstances. Such changes can occur in Ambulatory Assessment as the needs of patients vary and can alter over time. Such rule conflict is compounded by the simple complexity of potential run-time events.

To address rule conflict within a sensor network this paper draws upon an established body of *Feature Interaction* research. Specifically the paper extends FI techniques to deal with rule-based sensor networks where the implementation can be either embedded within the network, or operated on a stand-alone basis. The granularity and nature of such rules distinguishes this domain from the more traditional applications of FI techniques.

This work is motivated by the PAM project which constructed a rule-based sensor network. A rule-based sensor network was designed, developed, and deployed within homes in a set of small scale trials. The PAM project aims to provide support

* Corresponding author.
E-mail addresses: ehm@cs.stir.ac.uk (E. Magill), jesse.blum@nottingham.ac.uk (J. Blum).

http://dx.doi.org/10.1016/j.pmcj.2015.08.005

in the home for patients with Bipolar Disorder. As the support is away from a clinical setting and is in a natural setting it can be considered as an example of *Ambulatory Assessment* (AA).

In response to the trials the authors produced a networked rule model to investigate rule conflict and resolution in greater depth. As a result a novel approach to detecting and resolving rule conflicts was developed. The work is set within the context of home-based psychiatric Ambulatory Assessment. Rules have been developed to control the behaviours of devices in a sensor network being used for psychiatric AA applications. This produces examples of rule conflict that can be found within AA sensor networks and forms a basis of research to test the efficacy of the proposed detection and resolution approach.

1.1. Contributions

This paper describes a rule-based Wireless Sensor Network that can be programmed on the fly. This provides the high degree of personalisation required for AA. AA monitors patients in everyday settings away rather than in a clinical environment, and it is important to program each WSN to "fit" the patient. Using this, we make a number of contributions. Firstly, we provide a taxonomy of rules common to AA WSNs. We then demonstrate that the flexibility they offer can result in rule-conflict and we describe a number of ways that these rules can conflict. We then contribute a novel approach to detect and resolve such conflicts. The paper identifies a number of scenarios that have not been published before. While this novel approach builds on research for the Feature Interaction problem, it is distinctive as it can operate at a low level of granularity on small rule-sets rather than the larger software applications and services that appear within FI research. Furthermore, the approach addresses all 5 types of FI identified in the FI literature; while the literature offers approaches on four types, this approach also addresses the fifth type. It also extends the existing methods of detection for the four types to operate successfully in a rule-based system, and shows that the approach is scalable within the confines of likely AA-WSNs which are expected to have a modest size.

1.2. Article organisation

The next section (Section 2) describes an overview of AA, WSN and Feature Interaction literature. This is followed (Section 3) by a description of rule conflict in terms of an extended Feature Interaction taxonomy. The proposed detection technique is described in Section 4, along with an overview of the experimental test-bed that encompasses this approach. A resolution approach is described in Section 5, and Section 6 explains the test-bed environment in more detail in preparation for the presentation of the results in Section 7. Section 8 describes future work and concludes with Section 9.

2. Background and related work

Rule-based sensor networks are not in common use for AA systems and so a contribution of this paper is describing and justifying the distinctive nature of this domain for the subsequent Feature Interaction research. The section describes two sources that motivated the choice and development of AA-WSN features; firstly functionality cited in the AA literature, and secondly experiences from the PAM project. It also provides background material on rule oriented sensor networks and related work for conflict detection.

2.1. Ambulatory Assessment functionality

Ambulatory Assessment, as defined by [1],

> "comprises the use of field methods to assess the ongoing behavior, physiology, experience and environmental aspects of people in naturalistic or unconstrained settings. Ambulatory Assessment uses ecologically-valid tools to understand biopsychosocial processes as they unfold naturally in time and in context".

A major advantage of AA over the more traditional forms of assessment is reducing the reliance on retrospective recall [2]. AA based studies are also more general than the laboratory, as they have ecological relevance and can monitor dynamic situations through time. Other benefits include higher compliance rates than traditional paper-based surveys, improved survey structures, and rapid data access for researchers [3]. AA functionality supports micro-longitudinal studies by regularly sampling a range of parameters in a subject's life. This requires a core service that samples events over time and then stores the data. Sampling often employs momentary assessment where mobile devices such as PDAs are used to prompt subjects for answers to a set of questions. A number of additional features are also described in the AA literature; these include mechanisms to adjust prompting schedules, an ability to also collect physiological and environmental data, and a means of combining a range of triggers. Triggers may also reflect the state of a subject, or be the result of a change of context.

This paper draws on the literature [4] and Fischer [5] to provide insight into available AA functionality. It is worth noting that the paper does propose more extensive capabilities (based on wireless sensor networks) than is currently reported. For example consider three available systems: Experience Sampling Program (ESP) [6], Purdue Momentary Assessment Tool (PMAT) [7], and MyExperience [8]. These systems allow researchers to set up diary study protocols on Personal Digital

Assistants (PDA) and mobile phones, but only one of the systems (MyExperience) provides appropriate access to external sensors; and even this is limited to the mobile environment. Alarm-Net [9] is a wireless sensor network for assisted-living and residential monitoring that does capture and respond to data at home. However it does not support the mobile environment and does not have a rich questionnaire facility. This paper will assume all these capabilities being available in the one system. In addition it will also encompass functionality proposed in the AA literature.

Intille [10] presents a vision of Context Sensitive Ecological Momentary Assessment (CS-EMA). He assumes that incoming multimodal data streams are continuously collected, analysed, stored, and can trigger alterations to the collection process. Collection processes are automatically personalised to subjects and situations that they experience by combining data on subject location, physical activity, proximity to others, and self-reports on psychological state. Raw data, therefore, must be processed on-the-fly, in order to estimate subject activity. Subjects may still be prompted about their thoughts, feelings, and activities, but these prompts are contextualised from the physical information. Furthermore, questions can alter in response to historic data, or if the data deviate from statistical norms. Other features proposed include: complex question and answer flow, question aggregation, allowing subjects to specify when they should not be disturbed, precise time-based triggering for certain questions (including the capability for recurrence), randomisation of questioning, and bounded timing limits to querying.

System adaptation is necessary to meet CS-EMA challenges and it is assumed to be a necessary network capability. Limitations reported in existing AA literature state it is important to move from reliance on static timing and event mechanisms (such as sampling every hour), towards studies that are able to dynamically collect data from a variety of intermixed sources. For instance, Collins & Muraven [11] are concerned with an over-reliance concerning subject self-reporting, and the lack of handling control conditions. They report technological barriers to being able to enhance their self-report data with behavioural and physiological information. This, however, is already technologically feasible as reported by the authors [12] from their experiences in the Personalised Ambient Monitoring (PAM) project. They collected data streamed from a variety of sources for offline analysis, although it proved inappropriate for determining control conditions in a live environment.

2.2. Wireless sensor networks

Sensor networks are collections of devices that have a degree of data sensing, data processing, communications, and data storage capabilities. The capabilities are provided by a range of components such as communications transceivers, sensors, and ultra-small computing boards. The boards typically have power sources, memory, and some input or output connections. The sensors detect signals from physical phenomena (such as thermal, electrical, or magnetic radiation) and are converted into digital values.

The detailed design of a WSN is constrained by its particular application or setting. Many factors such as the power consumption of battery based devices, security, accuracy, reliability, costs and so on, must be balanced. WSNs have been employed in a wide range of medical, defence and industrial settings yet AA has not received a great deal of attention from the WSN community.

Sensor networks for AA require a high degree of transparency and personalisation, as well as minimising the number of devices used for monitoring. This is to help lower the subject burden and keep them and their care providers informed about how the subject is being monitored. Personalisation is an important factor in determining how a sensor network should be programmed. It must be possible to program the network behaviour for an individual, ideally on the fly to ensure the approach is tenable. From the perspective of this paper the way a WSN is programmed is important.

2.3. PAM project

Motivating the work presented in this paper on WSN rule conflict was PAM [13,14], a three year project across three universities investigating sensor technologies for people with Bipolar Disorder (BD) keen to control their disease with as little external medical intervention as possible. A particular effort was made to allow the network of sensors to be personalised and ambient, and operate potentially without the need for a centralised resource. While many projects have described sensor systems for general medical care or homecare, to the best of the authors' knowledge, the PAM project is the first to attempt to obtain activity signatures from the mentally ill using a network of environmental and worn sensors. This makes it a valuable source of suitable psychiatric AA features. A contribution of the PAM project is the introduction, within a mental health setting, of a rule-based approach embedded within the network. This permits dynamic and straightforward personalisation of network behaviour. It also supports additional equipment as and when it becomes available. The project constructed an infrastructure composed of off-the-shelf and custom-built wireless sensors, and a rule-oriented programming architecture to collect, process, and store data that can be related to models of care.

In a small technical trial within PAM, participants wore two matchbox-sized sensor nodes: a commercial sized *GPS* sensor; and a project-built device containing an accelerometer, a light sensor, and a sound level sensor. The devices use *Bluetooth* to communicate with their mobiles phones. The phones in turn pass the collected sensor data to the home PC for storage and analysis. The phone stores (buffers) the data until it is within range of the PC, when it can use *Bluetooth* to pass the data on. This provides a subject with a high degree of mobility as they need not stay at home or remain within wireless range of the PC.

A particular aspect of PAM was investigating the efficacy in a mental health setting of embedding rule engines within a sensor network. Given the processing capabilities of mobile phones it was natural as a first step to control the sensor data collection through the phone using a rule-orientated approach. In addition PAM used rules to personalise momentary assessment of subject activities and moods through on-phone questionnaires. To handle this rule-based programmability PAM-A was constructed to manage the storage and execution of rules within the network.

2.4. Programming WSNs

Currently, no consensus has been reached in the research community as to the best approach to programming sensor networks; an issue that is exasperated by the wide variety of network requirements. Rule-based middleware for sensor networks has been used in a number of projects. Sen & Cardell-Oliver describe [15] a rule-based approach as having simplified programming and concurrency models that make program correctness easier to prove, that they are power efficient, and that rule-based systems remain sufficiently expressive at high conceptual levels.

The EU-funded MONARCA project [16] has similar goals to PAM in supporting self-help for bipolar patients. It too employs phone-based questionnaires and merges this information with objective sensor data. Unlike PAM however, more use is made of physiological data and it does not rely on direct home environmental data. More importantly from the perspective of this paper, MONARCA does not employ a rule-based approach; this limits its degree of personalisation and flexibility as rule-based system can support run-time changes. The focus of this paper is to address the rule-conflicts that arise from rule-based approaches such as PAM.

According to Terfloth et al. [17] thinking about WSN from an event paradigm is more effective within sensor networks than the imperative paradigm. Rule orientation, they argue, is a more natural way to express programs for sensor networks. In addition, Fei shows [18] that application developers using rule-based middleware are protected from complexities arising from tight real-world integration, network dynamics, and resource limitations.

Given the role of rule-based sensor networks in AA, it is noteworthy that from a Feature Interaction perspective the controlling rules provide a much finer grained form of programming than seen in traditional Feature Interaction domains such as telephony. Rather than relatively self-contained large incremental features, the rules are smaller, often more numerous, and may alter within rather shorter time-frames than seen in more conventional systems. Although the telephony literature will quote the number of features being over 100, most telephony Feature Interaction papers address a much smaller number. This work explores bottom-up device programming that allows multiple providers to offer services and to form *ad hoc* reliable networks. The network can change over time with devices being added and removed as technologies change.

Given a rule-based approach it is important to understand the granularity of such control. Rules can be used to control sensor networks and can simplify the reasoning processes about network behaviour. The term *rules* is short for the term *production rules*, which were originally used to define formal grammars. In addition to programming sensor networks [17], rules have been used to program intelligent systems [19], and expert systems [20,21]. In these settings, rules are axioms of first-order predicate calculus used in connection with other axioms. They contain two parts: an antecedent and a consequent. These parts are connected within an inferential *if–then* framework such that: *if an antecedent is true then the consequent should also be inferred as true*.

2.5. Feature Interactions in rule-based sensor networks

The Feature Interaction problem is an active research area that addresses distributed software systems where the interacting software components are developed separately. Often the components are optional and are associated with distinct user discernible functionality. The field originates from the 1980s with the wide-scale introduction of programmable telephony equipment where marketable telephony features such as *three-way-calling* were sold as distinct items and were frequently developed by separate software teams. Feature Interaction forms the research context for the proposed rule conflict and resolution approach. This is appropriate as rule-sets in a WSN may originate as distinct items offering a particular functionality. The proposed approach extends Feature Interaction research to fit a rule-based sensor network. This section briefly describes the Feature Interaction problem, and explains the distinct nature of rule-based networks.

Feature Interactions are generally expressed in the literature in terms of scenarios rather than formal definitions. In that spirit consider a classic telephony example from the Feature Interaction literature that involves the user Alice who subscribes to the feature Originating Call Screening (OCS), which is configured to screen or block outgoing calls from Alice's telephone to the user Charlie. Another user Bob subscribes to the feature Call Forwarding when Busy (CFB), which ensures that any incoming calls received by Bob are forwarded to Charlie when Bob's telephone is busy. A Feature Interaction can occur if Alice calls Bob when he is busy, because either the call from Alice would be forwarded to Charlie, thereby invalidating her OCS feature, or else the call would be blocked invalidating the CFB feature. In either case, the operation of one of the two features would be changed by the presence of the other. Neither feature is in error. Both work fully according to their specifications. It is only when they share the control of resources that a conflict or Feature Interaction is seen. The goals of the two features conflict and are in effect incompatible.

Although the Feature Interaction problem was known in the industry for some years it was only later that the problem appeared in the research literature [22]. In time the FI research literature expanded to become concerned with a broader

range of system problems that emerge when services and features are combined within a system to produce unexpected results. FI is now witnessed in a wide range of system types, and is recognised in a wide variety of circumstances and domains (for example Internet applications [23], home automation [24–27], enterprise collaboration platforms [28], Web Services [29], and software product lines [30]). An example of the types of FI issues that can arise in such domains is the safety systems in smart cars that ensure that on impact the doors unlock, but thieves striking a car bumper or tampering with the impact sensor can potentially override the security system.

Many domains have a broader view of the term feature than seen in telephony yet retain the incremental nature of system behaviour. For example, a feature is defined by Calder et al. [31] as "…a component of additional functionality additional to the core body of software". There are other definitions, but they all try to express the way that one component of software can influence another in ways that are unexpected and unpredicted. This paper draws on this research as rule-sets executing across a WSN may offer conflicting functions too.

Although FI research informs the detection of rule conflicts in rule-based sensor networks the domain has a number of distinguishing aspects that require particular consideration. Programmable WSNs offer the homecare and ambulatory assessment settings the ability to personalise their monitoring and responses, and indeed these can change over time to respond to changing patient needs. This implies singular solutions that require a high degree of self-management. In addition the granularity of the rules is small which constrains the number of suitable FI approaches. Work by Marples [32] is suitable and has been extended by Wilson and Kolberg for the Home network setting which is highly relevant. However this work is incomplete and a particular class of Feature Interaction cannot be detected. This paper proposes a new approach that detects this class of problem to provide a more comprehensive self-management of WSNs.

2.6. Rule conflict in WSNs

Here we present an example without a formal syntax of rule conflict. Consider a small sensor network designed for a bipolar disorder patient for example consisting of three devices, each containing a distinct set of rules. The three devices might be a *location monitor* node (such as a wearable Global Positioning System (GPS) unit) to provide patient position, an *activity monitor* node to capture the activities the subject is performing (perhaps using a custom application on a mobile phone), and a *home monitor* node that collects information from a variety of sensors in the subject's home. Crucially the nodes must work cooperatively to combine the information to produce a useful reported outcome, hence the importance of the communication infrastructure.

The system might be programmed to generate notifications when a particular behaviour pattern alters in light of a patient location either inside or outside their home. So while rules can be distributed across the nodes to produce a particular system function, it is very likely that a number of system functions will be implemented within a particular sensor network. In this setting rules may originate from different sources. It is likely that a number of stakeholders will generate different sets of rules across a network. In other words the rules may well be written by different people who are not fully aware of the other rule authors. In addition each patient may simply have a unique set of rules given the desirability of personalising the system to individuals. So there is no single agency with responsibility for ensuring that the combined operation of the rules works coherently to the satisfaction of all the parties. In other words there is unlikely to be an effective *software development process* in place. It would not be tenable on this scale given the number, the variety, and the rate of change when such systems are deployed in earnest.

Consider the example in more detail. The device behaviours are controlled through rules on each of the devices. For example the *home monitor* may have a rule that stipulates the device should only be active when the subject is located at home, since the subject lives with other people. The *activity monitor* may have a rule stating that when the subject is at home the *activity monitor* should be limited to only selecting from relevant home activities. Assume a second rule on the *activity monitor* dictates that if the subject is performing the activity *travelling*, then only the start and end locations of the journey should be recorded to conserve resources. Finally, assume another rule governs the length of the *travelling* activity. This may be set to ensure that if the subject enters the car, drives, makes a *micro stop* (such as buying milk) and arrives at a *macro stop* (for instance a place of work), it is all considered as part of the same travelling activity. So when the subject travels from home, returns home briefly, then sets off again, the *home monitor* is deactivated and the other two nodes enter their travelling states. When the subject returns home briefly (a behaviour not initially considered by the researchers), the *home monitor* remains off (a turn-on trigger is not sent by the *activity monitor*) and does not capture any further abnormal behaviour patterns, resulting in a loss of useful data. A rule conflict has occurred.

The particular focus of this paper is providing a means to ensure the rules are not in conflict. That is, that the goals of one rule, or set of rules, do not interfere or conflict with those of another. While this does not fully address the need for a comprehensive validation of the rules, it does remove the more glaring inconsistencies.

Conflicts in rule-based sensor networks are established in the literature. Often this takes the form of rules (or policies) in the Event–Condition–Action format. Solutions are offered employing Resolution Policies [33], Event Calculus [34], and Detection Algorithms [35–37]. P. Carreira et al. also discuss conflict detection and resolution for ambient intelligent systems based on environment variable constraints [38]. In contrast this paper focuses on situations where rules are employed and uses the Event Calculus to reason about conflicts in sensor network device rules. This is the first contribution to Ambulatory Assessment (AA) WSN conflict assessment and resolution. AA requires a high degree of self-management and the paper reports a more comprehensive coverage of conflicts than competing schemes.

Table 1
Feature rules tested for conflicts.

Outbound Data Screening (ODS)
Context Triggering System (CTS)
Data Storage Through Processing (DSTP)
Data Storage Unconditional (DSU)
Data Recording Frequency (DRF)
Do Not Notify Unconditional (DNNU)
Inbound Data Screening (IDS)
Retry Data Transfer On Unavailable (RDTOU)
State Triggering System (STS)
Prompt
Time Synchronisation (TS)
Automatic Data Transfer (ADT)
Redirect Data Stream (RDS)
Report Location (RL)
Activate Immediate (AI)

Much Feature Interaction (FI) research assumes a relatively well resourced context. So although scalability is an issue in such research, limited processing and communication resources are of less concern. The need for portable and ideally connected devices to provide AA capabilities makes Wireless Sensor Networks (WSNs) an obvious approach. Here, however, computing and communication capabilities are frequently more limited than other FI domains. In an AA setting it is also important to limit the burden on patients; and so it is useful to minimise the number and size of sensors used. By tuning the number and type of sensors for individual patients, it is possible to constrain this burden. Such personalisation ensures that patients only use devices that are truly required. Programmability of WSNs makes this a possibility.

Personalisation also allows the WSN to be programmed to an individual's clinical needs and to change for the patient as their needs alter in time. So the ability to program on the fly is also important. Given the resource constraints a rule-based approach is advocated. This is a dynamic AA environment employing low level rules to provide functionality. This has three implications. Firstly any rule conflict mechanism must be reliable. All known types of FI need to be addressed given the AA setting. In addition the dynamic nature requires a high degree of self-management. So again the coverage of any FI management must be comprehensive to permit automatous operation. Secondly, any FI approaches used must be able to deal with a low-level of programmability. Here small rule-sets are proposed offering a rather small programming granularity. In most FI settings the granularity is larger. Whole programs and sizeable modules are used, where often a particular (user) function is implemented in one indivisible software component. Here in contrast the functionality is spread across many rules. This limits the number of FI approaches that are suitable. Earlier it was noted that work by Marples [32] is suitable and that it has been adapted to work in a network setting. This thread of research matches our requirements well. However it was noted that this work is incomplete and that a particular class of Feature Interaction cannot be detected. It is argued that in this AA setting any FI approach used must be more comprehensive. Hence this paper proposes a new approach that offers a more complete self-management of WSNs. Thirdly, the lack of resources also requires FI approaches that are more efficient. So in addition to being comprehensive, they must not require significant processing or inter-process messaging. Hence this paper proposes an approach that has limited resource implication.

2.7. Feature rules

Given these two sources of realistic psychiatric AA features, the 17 feature rules in Table 1 were developed (see [39] Chapter 4 for rules descriptions). The rules have been grouped into two categories: device control rules and knowledge rules. Device control rules handle the way that devices sense and manage data, handle programming updates, and interact with subjects. Knowledge rules are used in the examination of the data to determine the states and contexts of the subjects, along with being used in the analysis of the quality of the data. A number of rules are presented below for both of these categories. The explanations are weaved into presentations of scenarios giving rule conflict in this AA setting.

Device control rules are used to manage how devices operate to collect and manage data, and interact with users. These rules are at the core of being able to collect, store and provide access to physiological and environmental data, as well as to present information to subjects, and prompt them for responses.

The rules presented in this work provide greater support for AA protocol designers than previous systems such as Experience Sampling Program (ESP) and Purdue Momentary Assessment Tool (PMAT). In particular as:

- Studies can mix event-based, time-based, subject state-based and contextual triggers.
- Subjects may be prompted for questionnaire assessment in dynamic response to the concurrent collection of on-body and environmental data.
- Thorough data handling and provision mechanisms.
- Rule-based presentation of information.
- Automatic conflict detection and resolution.

These advantages aid the uptake of AA. Larsen [40], for instance, pointed out the primacy of research questions over data collection and analysis. The above items provide researchers with greater abilities to pursue their questions, and allow them to be flexible in their approaches to their collection and analysis methods. Furthermore, these points lead strongly towards realising Intille's CS-EMA + 10 vision [10] of AA future developments, including supporting the data collection procedures he outlined for longitudinal studies such as data-reactive subject prompting.

The literature provides some support for the selection of features and rules, and indeed priorities, employed within the experimentation. The rules were selected following a review of the literature on AA and Feature Interaction (FI). In addition experience from the Personalised Ambient Monitoring (PAM) project influenced the choices made. While not exhaustive this did provide representative rules for care settings. The rules were translated from literature-based descriptions into diagrammatic and EC based forms. In only a very few cases (such as for TS) did the descriptions in the literature include working algorithms. So interpretations of the descriptions were necessary. While attempting to represent the rules accurately in EC rule forms, alternative interpretations are possible such as having multiple forms of the rule STS. In contrast, other approaches such as Calder et al. [41] are able to analyse rules directly from log files of running systems.

3. Rule conflict

Using the 17 rule-sets it is possible to illustrate a number of rule conflict scenarios. This is not exhaustive. Exhaustive checking is described later in the results (Section 7); here selective cases are presented to provide insight into the types of conflict that occur. It also provides a description of nine of the 17 rules.

A taxonomy is essential to get to grips with Feature Interaction. In particular as it captures the nature of the exchange of events between devices that lead to a rule conflict. This paper uses the extended form of Marples' FI taxonomy described by Wilson et al. [24]. It consists of five types of interactions: *Shared Trigger Interaction* (STI), *Sequential Action Interaction* (SAI), *Looping Interaction* (LI), *Multiple Action Interaction* (MAI) and *Missed Trigger Interaction* (MTI). Each is considered in turn. These conflict types derive from telephony but have also been considered in alternative domains such as home automation [24]. Here, we consider their implications for rule-based WSNs with AA-like characteristics. Some of the examples are within a single device while others are between rules in separate devices. While *multiple component* and *single component* interactions are important aspects in telephony Feature Interaction, particularly from a detection and resolution perspective, here the difference is less pronounced.

The contribution here is in both applying this taxonomy in a rule-based domain, and extending the work reported in the literature to work in a rule-based environment. Importantly the latter required developing a new algorithm to ensure MTI interactions could be detected, as earlier work from the literature had failed to do so. However the adaptation of existing algorithms for the other four types to a rule based setting is also novel. The former contribution relates to showing that the taxonomy does indeed apply in a WSN rule-based setting and developing a wide range of scenarios in the process. More generally the identification and classification of conflicts in an AA setting is new and useful.

3.1. Shared Trigger Interactions

A Shared Trigger Interaction (STI) occurs when more than one rule or rule-set responds to the same trigger, and the response of any of the rules is altered by the triggering of any of the other rules. In terms of the rule forms described in Section 2.4, STI can be defined as the antecedents of multiple features being satisfied such that they each perform actions in response to the same triggering event, and the operation of one or more of the features is different from how it would have reacted had it been the sole responder. So in short, STI conflicts are caused by multiple rules reacting to the same trigger, and one or more of the resulting actions differ from how they would have been, had only one rule responded to the trigger.

The following example is characterised by the use of rules that run on the same device although other scenarios involve more than one device. A mobile phone has subscribed for the *Context Triggering System* or CTS feature. This feature is a rule set that allows a subscribing device to respond to changes in contextual information and can be regarded as belonging to the Knowledge rules class. CTS monitors contextual information specified by the mobile phone and when it is within the bounds set by the phone, it informs the phone. The mobile phone listens on an agreed channel and can take action when informed that the context data is within bounds. The phone, acting as the subscriber, can check the data is truly within bounds and if so the phone can carry out its preferred action. This type of feature can be useful for triggering responses to changes in the environment. In this example the phone also subscribes to the *State Triggering System* or STS feature. The STS rule set allows a subscribing device (in this case the phone) to respond to changes in state information and so is also a Knowledge class rule set. Again the phone listens on an agreed channel and is informed if its selected state variable is within its chosen bounds. Thus triggered, the phone can check the variable is indeed within bounds and if so the phone can execute its preferred action. This type of rule set is useful for responding to changes in subject state.

In the example the CTS feature has rules to trigger the phone when the subject leaves the home, in other words the context is no longer the home. In response, the phone has rules to prompt the subject to ask what activity they are doing. Similarly, rules have been set up using STS to trigger the phone when the subject's behavioural state changes from sitting to walking. In response the phone is programmed to prompt the subject about their emotional state. A conflict can occur when the subject walks away from home. Here, both prompting rules are triggered, however only one can prompt for a response

to its question. In other words, this is a STI conflict as one set of rules will not result in a prompt so its response has been altered by the presence of another feature.

3.2. Sequential Action Interactions

Sequential Action Interactions (SAI) occur if a rule is triggered in response to the actions of another. Side-effects of these conflicts are not always harmful and can be beneficial, however the example described here demonstrates both a positive and a negative outcome.

In the above example a phone subscribes to the STS feature described above. Here the rules ensure that if the subject's heart rate exceeds a particular threshold, then the capture rate of the accelerometer in the phone is adjusted. This is achieved as the phone also subscribes to the *Data Recording Frequency* or DRF feature. This Device Control rule set determines how often a particular device, such as a sensor, will collect data. Here the DRF feature can adjust the sample rate from the accelerometer. Under normal conditions STS will signal DRF and the features will work successfully in tandem as expected, and so this is a benign example of SAI. However if the phone also subscribes to CTS, then it is possible that a race condition could arise such that it would be uncertain whether the actions of STS would be performed, and therefore DRF would also be uncertain of firing. This last aspect of the scenario is a negative example of SAI.

3.3. Looping Interactions

Looping Interactions (LI) or conflicts occur when rules are chained such that the triggering of one of them leads to the triggering of another, and this in turn leads to a sequence where the original rule is triggered again. According to Wilson [42], LI are often considered as special cases of SAI where the chain of features is closed to form a loop and the operation of the chained features leads to a redundant cycle. He argues that while SAI can potentially be beneficial, LI cannot.

In the example above two rules are used on different devices. Both features are Device Control rules that alter the management of data. A mobile phone has the *Automatic Data Transfer* or ADT feature set-up to automatically send data to the home gateway. The home gateway in turn has the *Redirect Data Stream* or RDS feature set up to redirect data streams to the mobile phone. This, for instance, could be necessary if a home gateway is partially offline for repairs. This configuration of rules across devices leads to a loop when the mobile phone attempts to transfer data to the home gateway which automatically returns the sent data.

3.4. Multiple Action Interactions

Multiple Action Interactions (MAI) occur when different rules attempt to control the same device at the same time. If multiple features attempt to provide instructions for the same device, then the features are said to interact by way of MAI. In some cases the interaction may be benign, such as when both services send the same instruction to the device. Alternatively the interaction may be intolerable, such as when the services send conflicting instructions.

The example above has two rules running on the same device that both help to manage data. The first feature *Data Storage Unconditional* (DSU) allows the receiver of a new incoming data stream to store the raw data, either directly to a file or to a database. DSU is the basic data management rule available in most AA solutions. In contrast in the second feature *Data Storage Through Processing* (DSTP) data is streamed, collected, and then processed applying appropriate algorithms once the streaming stage is over.

If a mobile phone subscribes to both features then DSU will store the raw data, while the other will process the raw data and store the result. This result may be benign if both are storing to different locations. If, however, the storage location is the same for both rules, and metadata are not used to differentiate raw and processed data, then this conflict will corrupt the data store. It is possible that data may be sent to the same location multiple times or, worse, data may be written by the execution of the first rule and then overwritten by the second one. The problems are compounded because race conditions may arise affecting the order data is written.

3.5. Missed Trigger Interactions

Missed Trigger Interactions (MTI) occur when one rule operates in a manner that prevents the triggering of a second one. MTI detection and resolution have escaped previous online and hybrid approaches to resolve FI. The closest result in the literature was made by Kolberg [43]. He reported that MTI was theoretically detectable, but it was not detected at run time owing to limitations in the underlying system architecture used for his experiments.

In the above example a prompting feature and a privacy feature are operating on the same device. The *Prompt* rule set aids what Intille [10] called Context Sensitive Ecological Momentary Assessment (CS-EMA), and is seen as an important aspect of new AA systems. The Prompt rule set allows researchers to solicit responses to questions from subjects, which can be chained together to form questionnaires for in-moment surveys. The *Do Not Notify Unconditional* or DNNU feature prevents a subscribing device from receiving notifications. If a mobile phone subscribes to DNNU then when the DNNU feature becomes active it prevents the Prompt feature from being triggered.

3.6. Exhaustive experimentation

The goal of this section (Section 3) has been to highlight the general potential for Feature Interactions in rule based WSNs, and in particular within an AA setting. The adopted features are based tightly upon the literature on AA functionality, both that within current systems and that predicted by authors in the field. These have been classified and through manual manipulation a number of representative scenarios have been presented. Given the genuine potential for such interactions in the future, a model has been built capturing the feature rules identified through the literature. In addition a series of algorithms have been developed and implemented within the test model that automatically detect and resolve rule conflicts. The next sections (Sections 4 and 5) describe the model and its automatic detection and resolution algorithms.

4. CLIPPER

CoLlaborative Information Processing Protocol and Extended Runtime (CLIPPER) provides a framework for novel rule-conflict detection and resolution algorithms. It employs the 17 AA rules and attempts to detect and resolve the five conflict classes described in Section 3. A number of techniques are used for the analysis, conflict detection, and resolution of rule conflicts. Subsequent sections present the evaluation of these methods. The contribution here is the description of an approach to detecting and resolving five forms of rule conflict within Ambulatory Assessment (AA) sensor networks.

4.1. Event Calculus rules

CLIPPER looks for different forms of conflict between collections of device rules by providing an environment that triggers all the devices rules. This operates in a controlled manner and the rules are exercised in pairs and are stimulated by a set of messages. The resulting rule execution sequences are then analysed for potential conflicts. The analysis mechanism employed differs for each particular form of rule conflict. The environment written in *Prolog* ignores the contents of the messages and the semantic meanings of the rules.

CLIPPER requires that the syntactic structure of testable rules and the rule clauses are presented in *Event Calculus* (EC) format [44]. The EC consists of three main concepts: fluents, actions (or events), and time points. Fluents are properties of the universe of discourse that can change in time. These properties may either take a form such as *the subject is in the house* or a quantifiable form, for instance the level of ambient sound in a room. A fluent can hold at a given point in time if it was previously initiated by an action and has not been subsequently terminated. Actions occur at points in time and can modify fluents. Time points provide a narrative based structure independent of any particular action.

Rules for CLIPPER analysis contain the following:

1. Domain dependent sentences (Σ) describe the effects of actions on fluents at given time points.
2. Domain independent axioms (EC) are the backbone of the EC. These are used to reason about whether or not fluents hold at given time points.
3. A narrative (Δ) provides a list of the events that occur and defines the temporal ordering (such as, given time points T1 and T2, T1 occurred before T2).
4. An initial situation (Δ_0) describes the fluents at the earliest time point.
5. Unique names (Ω) for fluents and actions.
6. A goal (Γ) is queried to determine whether a fluent holds at a given time point. A goal Γ is modelled by $\Sigma \wedge (\Delta \wedge \Delta_0) \wedge EC \wedge \Omega$.

CLIPPER begins by loading the device rules and then proceeds to check for conflicts between each pair of rules (including checking rules against themselves). Checking a pair of rules involves two phases: initialisation and detection.

The initialisation phase resets the Prolog environment by removing all assertions from it. It then adds a number of time points (establishing a linear order amongst them) and initialises a fluent that represents a message that can be sent to the rules.

The detection phase involves passing device rules, time points and messages to conflict detection rules. The conflict detection rules are then used to evaluate whether or not the device rules are concordant or conflict, and to record the evaluation results.

4.2. Rule format

The rules describing device operation are written using the Prolog rule syntax and are composed of a head followed by conditions containing initial situation (Δ_0) sentences, domain dependent (Σ) sentences, and narrative (Δ) sentences. The rule heads are limited to the form *rule-name (Trigger,T)*, where *rule-name* is the name of the device rule, *Trigger* is the triggering message and T is the initial time point. Although Trigger and T are the only arguments of the rule, it is possible that additional variables are used within the rule. These variables may be instantiated by assertions from other rules or the triggering environment. This allows the rules to be written in a generic format with variables that can be instantiated as appropriate for the device and its context, with a similar outcome as Blair and Turner [45] policy variables. Δ_0 sentences can be used to describe initial state constraints. These terms may be useful in cases where rules are only activated when

they are in particular states. For instance, the activation of screening in *Inbound Data Screening* (IDS) requires that the sender (Alice) be on the screening list of the recipient (Bob). This fact can be asserted into the initial situation as: *Initially-p(bob-screen(alice))*.

Σ sentences describe actions that may be taken by rules. For example processing data can occur as a result of the *Data Storage Through Processing* or DSTP feature described in an earlier section. This can be described in a Σ sentence as: *Initiates(receive-data,process(data),T2)*.

Δ sentences describe the activity sequences of the rules. A sender attempting to connect to a recipient device, for example, is an activity of IDS. This can be described in the terminology as: *Happens(sender-attempts-connection,T1)*.

4.3. CLIPPER analysis

The analysis components of CLIPPER sequentially compare all permutations of pairs of rules. To analyse each pair of rules, CLIPPER carries out an initialisation process and sets up the rules with the correct message and time values. It then uses given conflict rules to check for conflicts between the device rules, and records the results. This step is repeated until no more rule permutations remain, when a completion message is written and the initial goal is satisfied.

Conflicts can occur that involve more than 2 rules. This reflects experience more generally in Feature Interaction research. *N*-way Feature Interactions are reported in other domains but tend to be rare and are often not addressed in the literature. An *n*-way interaction strictly only occurs if all *n* features need to be present; if two are interacting within a larger set, then it is deemed a pair-wise interaction if it can be reproduced with the pair alone. This approach, like many in the literature, addresses pair-wise interactions only. This captures the vast majority of conflicts.

Events and the situations they arise from are not always perfectly known in the real world. Contexts within areas such as CS-EMA are often inferred and determined to be at a level of fuzziness or probability. For instance, given a particular configuration of sensors and their readings in a home, perhaps there is only a chance that a particular state of the monitored individual is known (for example given the light, sound and motion sensors, there may be an 80% chance that the subject is awake, and a 20% chance they are asleep). To handle such probabilities and fuzziness using the rule format above, one can stipulate multiple rules for each of the possible outcomes each containing variations in the Δ sentences, and then have a master rule with the probabilities of firing them. Such an approach then allows CLIPPER to perform conflict assessment on each of the rules.

4.4. Conflict rule notation

The conflict detection rules are written using EC goal (Γ) sentences along with additional clauses for ordering test sequences and writing output. The sentences are used to ensure that particular fluents hold at given points in time. The fluents that should hold are dependent on the nature of the interactions being analysed. MTI analysis, for instance, uses sentences to check whether the contents of a message continue to exist when they are passed between rules. For example, a sentence can be written as: *holdsAt(message(Content),T2)*.

In order to perform conflict detection, CLIPPER requires conflict rules and device rules in addition to the core rules described above. Five forms of conflict analysis are described, one for each class of conflict. Code listings for each of these algorithms is given in [39] Appendix A. Each algorithm operates independently from the others, but each relies on the predicates from the Event Calculus (such as *holdsAt*, and *before*).

4.5. Missed Trigger detection

As shown in Fig. 1, detecting MTI can be accomplished by testing rules sequentially to ensure that a common fluent holds before being passed to each of the tested rules. The fluent can be considered as a type of triggering message that should remain in a consistent state between rules. Such an approach need not make any assumptions about the contents of the message, nor about the actions that should be performed by the rules, nor about what the rules do upon receiving a message. This check is captured in a MTI concordance rule. Feature description rules satisfy two qualities: successfulness and destructiveness. If a rule does complete all its predicates it is successful, otherwise it fails. If a rule modifies or absorbs an incoming message it is destructive. Modifying a message results in its termination at a point in time, and initiating a new message at either the same time point or later. So a destructive rule terminates a message, and in contrast a passive rule does not. Rules can therefore be described as belonging to one of these types: Passive–Failure, Passive–Successful, Destructive–Failure, and Destructive–Successful. Feature rules conflict by MTI if the initial rule is a destructive one.

4.6. Shared Trigger detection

Shared Trigger Interaction (STI) occurs when multiple actions are performed in response to the same triggering event, and the behaviour of one or more of the actions is different from the behaviour arising from a single rule response to the trigger. Testing for STI, shown in Fig. 2, can be accomplished by querying a feature rule, then resetting the query environment, querying a second rule, then querying the first rule again. If there are differences in the Σ sentences between the first and second instances of the first rule then the first and second rule conflict by STI.

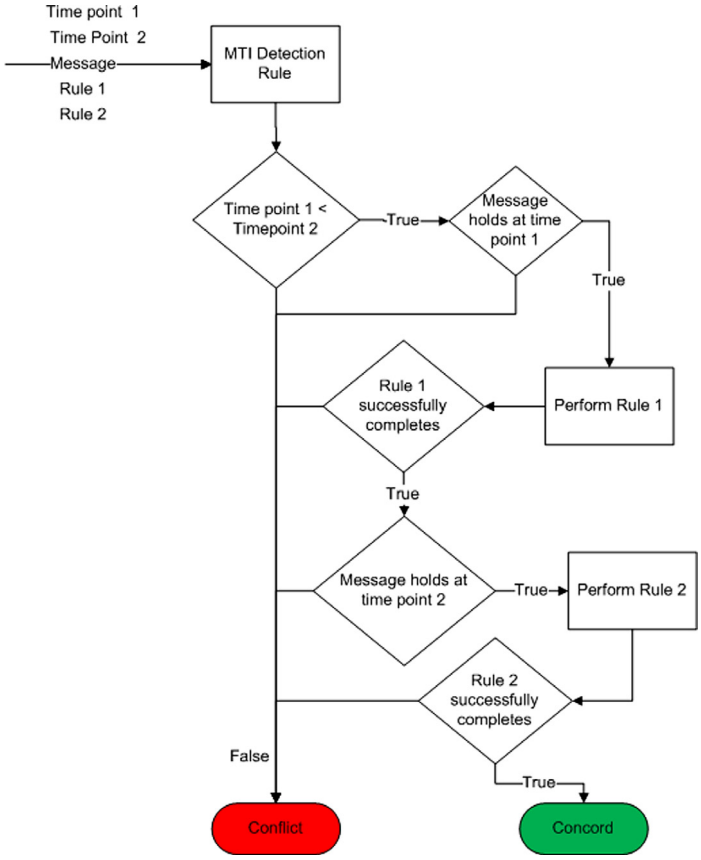

Fig. 1. MTI detection algorithm diagram.

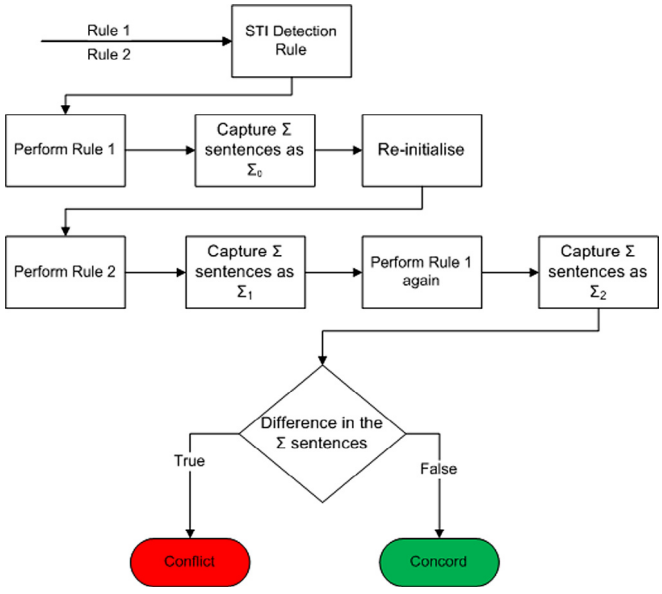

Fig. 2. STI detection algorithm diagram.

4.7. Multiple Action detection

The occurrence of Multiple Action Interaction (MAI) is caused by the attempted control of a single device by multiple rules at the same time. The given approach does not distinguish device types and has no knowledge of where the rules are

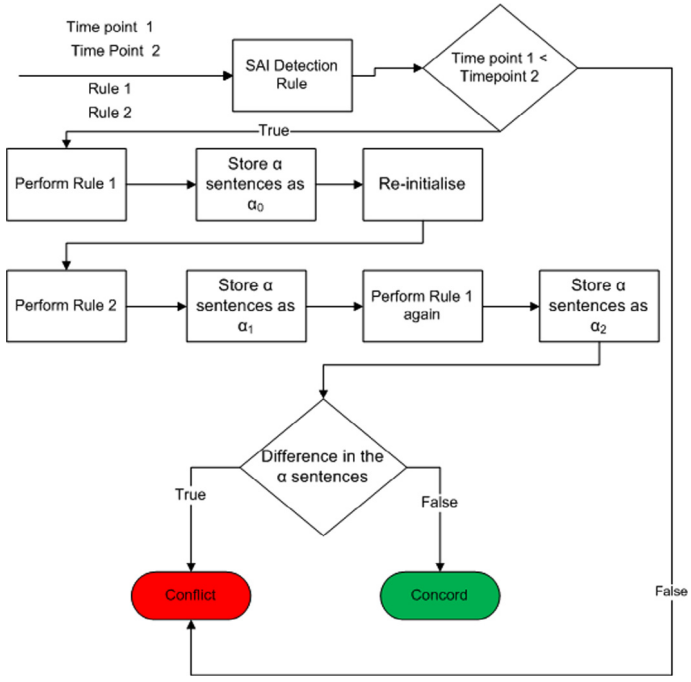

Fig. 3. SAI detection algorithm diagram.

triggered. Adding such information, although possible, will yield little benefit as it amounts to a declaration that the rules are triggered on the same device. If such a declaration is desirable, then one only needs to assume that the rules are triggered on the same device. Given that assumption, MAI can be detected using the STI detection rules.

4.8. Sequential Action detection

A shown in Fig. 3, SAI can be detected by testing if a feature rule performs an action that leads to the performance of actions by a second rule.

4.9. Loop detection

LI occurs when one rule triggers another which in turn causes the first one to be re-triggered. LI, therefore, is a special case of SAI that can be defined as SAI leading to the triggering of the first rule's actions. This can be detected by performing SAI checks on the rules and examining the output for cases where two rules have SAI regardless of whether they are the first or second rule. For SAI to be detected a rule must trigger a change in behaviour of another at a given point in time. The time points and the rules are called by the framework which then checks for differences in the way that a rule executes, whether it runs before or after another rule. If its operation is the same in both cases then SAI is not detected.

5. Resolution

Having shown that it is possible to detect conflicts, it is now important to consider what may be done about them. This section describes strategies that may be applied to deal with the conflicts. It also describes how the decisions of which to use can be reached and how these can be applied to cases of the five interaction types previously discussed.

5.1. Device priorities and CLIPPER

CLIPPER uses a rule-based device priority system to handle conflict resolution. The resolution rules determine which of the conflicting rules have precedence, and which are to be temporarily disabled in order to avoid the interaction. Device and network-wide priorities are combined to determine rule precedence. Device level and system-wide priorities are described as facts that are taken into account during resolution. These priorities are guiding principles for each device.

Device priorities have a considerable benefit in handling the heterogeneity of sensor networks. For example, the priorities of a gateway device that is drawing power from the home and has a high-speed Internet connection will be considerably different to those of a mobile phone with limited storage capacity and bandwidth. The mobile phone may prioritise data

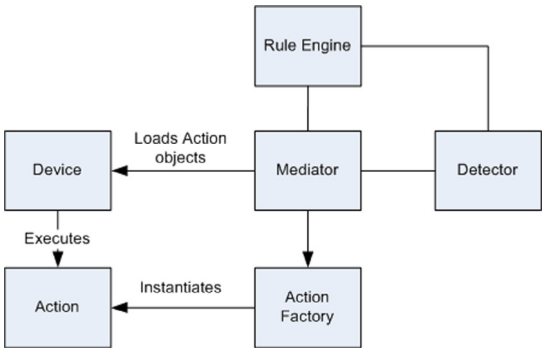

Fig. 4. Overview of CLIPPER architecture.

processing, whereas the gateway may prioritise data integrity since it has ample power and data storage facilities. If the MAI interacting rules for Data Storage Unconditional (DSU) and DSTP are present on each device then the appropriate resolution for the wearable device is to disable the DSU feature, but the appropriate resolution for the gateway is to disable the DSTP feature. For the resolution system to decide, the mobile phone administrator could set a device level priority for data integrity to a high value and so set DSTP to a higher priority than DSU. The gateway administrator can make similar a priori decisions for the gateway priorities but giving a higher priority to DSU. The outcome would therefore depend on the device in question and the chosen priorities.

In this approach the resolution system always disables rules with the lowest priority. Multiple rules with the same lowest priority (including no priority) are all disabled. This allows a system to gracefully handle conditions that can occur where priorities are not present or do not pertain to the interacting rules. In some cases, such as the LI example in Section 3.3, the interactions may occur between rules across devices and this may result in the features having the same priority. In that example it is quite possible that the mobile phone has transference of data as its highest priority and the home gateway has security as its highest priority.

The priority system of CLIPPER works by consulting the priority files associated with each device, the core conflict analysis and resolution files and the conflict detection report file. Variables from each of the conflicts found in the conflict report are passed in to the resolution system's resolve rule. The variables include the type of conflict (such as MAI) and each device and device rule in conflict. Each rule is checked to see if a priority has been applied to them. If both have priorities they are checked to see which has precedence. The rule with the lowest precedence is disabled as are rules that have no priorities applied to them. If two rules have equal precedence then they are both disabled.

6. Architectural introduction to CLIPPER

CLIPPER is used in this research to simulate how network devices can behave when influenced by conflict detection. The focus of CLIPPER is on the combination of rules, and abstracting away network details. Rather than network size, it is the number of differing rules in the network which influence factors related to conflict analysis, particular the amount of time such analysis takes. This approach uses a mediator to analyse device rules and load appropriate actions into devices. CLIPPER-based applications run on physical devices using key concepts defined for devices, mediators, their rule engines, and actions performed by the devices. The devices are controlled through logic rules and facts. Actions, rules, and facts are loaded and unloaded into devices and knowledge bases at run-time. This allows networks of devices running CLIPPER applications to change and react to new information.

Fig. 4 shows an overview of the main architectural elements of simulation applications which include:

- *Device* classes represent physical devices such as mobile phones and gateways. Multiple *Devices* can be added to an application. *Devices* are provided with a path to the files containing the rules that specifying their behaviour. Devices are loaded with *Actions* by *Mediators* and execute them in sequence.
- A *Mediator* loads device rules into a *Rule Engine*, passes the rules to a conflict *Detector*, and loads *Actions* into *Devices*.
- *Rule Engines* query declarative logic facts and rules given goals.
- *Detectors* perform conflict detection by loading the device rules files and CLIPPER core analysis logic files into a *Rule Engine* and querying it for a conflict goal.
- *Action* classes implement EC actions (or events) from device rules into *Action* objects. For instance, a rule for *Automatic Data Transfer* (ADT) might have a happens sentence which includes an action for *attempt-connection*. An *attempt-connection* action class is required to handle the connection attempt over a particular network protocol. *Actions* are executed by their containing Devices.
- *Action Factories* reify *Action* classes given a complementary EC rule name supplied by a *Mediator*.

In CLIPPER, device objects observe and react to stimuli input into a physical device. Device objects are programmed to watch for input into the physical device. They register with mediators and report to them when new data enter the system. Users can add new devices, or change device behaviour, by registering with a mediator. This requires three steps:

1. Identify existing action classes, and/or program new ones and register them with the action factory.
2. Register the device with a mediator.
3. Add rule sets regarding the device and actions it should perform to the rule set.

Rule sets are declarative files in the format described in Section 4.2 that pertain to a given device. Actions are Java classes that extend the Action interface and are named to correspond with the alpha parts of Δ sentences. Rule sets and action classes are programmed by device users (where a user may be someone performing the monitoring and/or being monitored) and made available for use by a mediator.

As described earlier, *Mediators* ensure that rule conflict is checked on device registration and removal. Mediators do so by passing all known device rule file paths to their detectors. The rule sets may be distributed across various devices, but the detectors will load all of them centrally for conflict analysis. The detectors detect conflict by preparing a temporary directory with the CLIPPER core analysis logic files, a logic file that it generates containing the files to consult, and a generated rule listing the device rules to analyse. Detectors load the generated file into a rule engine which interfaces with SWI-Prolog using the JPL library. The rule engine uses Prolog to consult the analysis and device rules files and then perform conflict detection. Conflicts are written as facts to a detection report file describing the type of conflict (such as MTI), and the pair of conflicting rules.

Mediators control the system-level logic to ascertain what system-wide actions should take place in response to an input. The mediators also control the inter-device communication. When devices are updated (such as when new readings are sensed) they notify their mediators of occurrences of events. Mediators then make calls to rule engines to determine the actions to be carried out. These actions are loaded into devices for execution. Conflict resolution should be performed by a mediator after it has checked for conflict. It should only load the actions of non-conflicting rules, or rules that have higher priority over rules of lower priority that conflict. To do so, it can use the device priority logic system described above with the conflict detection report file and device priorities files to ascertain the appropriate actions to load into devices, ignoring the low priority conflicting ones.

Rule engine objects query knowledge base facts and rules. Satisfied queries return the names and parameters of actions to perform, along with the identities of devices to perform the actions. These queries return from the rule engine to the calling mediators. The mediators use action factory objects to generate action objects which are loaded into the appropriate devices. The enqueued actions are then executed by the devices. Rules, facts and actions can be loaded and unloaded into the rule set and action factory at run-time. This allows actions to load new facts and rules into the rule set.

A standard usage sequence is depicted in Fig. 5. In the sequence diagram two devices register with a mediator. Sometime later, one of the devices sends an update message to the mediator. The mediator's rule engine is called to determine the actions that should be performed. A given action is then reified by the action factory. The action is passed back to the mediator which loads the action into the device that should perform it. The action is then performed by the device, ending the sequence. In this diagram, the number of devices shown, along with the decision to have the first one generate the update message and the second one perform the action are arbitrary.

An example using such a sequence involves a mobile phone and a wearable GPS unit. The phone can act both as a mediator and as a device. When the wearable GPS unit sends new data to the mobile phone, it can call rules to determine actions to take based on the current location. An action such as one to ask a question about the subject's current activity can then be returned by the rule engine to the mediator on the phone which may then be loaded into a questionnaire application on the phone.

7. Simulation results

By running CLIPPER within a test harness it has been possible to show that conflicts do indeed emerge and can be detected and resolved automatically. 17 features were employed in the simulation runs; 9 described earlier[1] and another 8 which are not fully explained here owing to a lack of space. The detailed results from running a particular scenario is described first, followed by a summary of the results from all the examples met in Section 3. Finally the result for a wider range of features is presented. The resolution proposals are based on a set of priorities configured for the resolution system. These were set by the authors based on likely settings for all of the rules used in the examples.

7.1. Conflict detection

To understand the simulation of each rule pair, consider the Shared Trigger Interaction (STI) example described earlier in Section 3.1. Here the two rules are *Context Triggering System* (CTS) and *State Triggering System* (STS). The CTS rule has Δ

[1] As two versions of the STS feature were used.

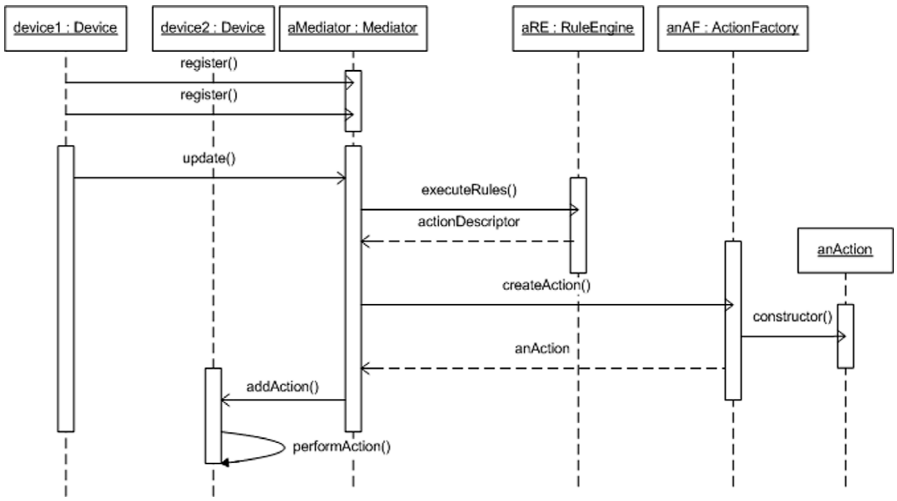

Fig. 5. Typical CLIPPER usage sequence.

Table 2
STI detection results.

	Rule 1	Rule 2	Result	Resolution (D)
1	CTS	CTS	STI	Both
2	CTS	STS (form A)	STI	STS (form A)
3	CTS	STS (form B)	Concordance	
4	STS (form A)	CTS	STI	STS (form A)
5	STS (form A)	STS (form A)	STI	Both
6	STS (form A)	STS (form B)	Concordance	
7	STS (form B)	CTS	Concordance	
8	STS (form B)	STS (form A)	Concordance	
9	STS (form B)	STS (form B)	Concordance	

Table 3
Analysis results for STI example.

Conflict type	Rules under test	Detected?	Resolved
STI	CTS & STS (A)	Yes	Yes

sentences that describe its activities. It also has Σ sentences that are activated if the message fluent holds at the time point following rule activation. This is a destructive rule because the message fluent is terminated at the activation time point by this rule. Two forms of the STS rule are presented in the results. Normally, only one form of a rule is used, however here two are given to highlight different ways the Σ sentences can be expressed. The first form of the rule (form A) depicts an active response to changes in data. To save space form A calls the CTS rule since the two rules are similar in nature. The second form (form B) uses the same Δ sentences as the other two rules, but is passive and does not have any Σ sentences. The two forms of the rule produce different results. The results are presented in Table 2 for every permutation of the features. The table shows that form B of STS is in concordance with the other rules, but the other rules conflict with themselves and each other. Rows two and four show the rules conflict with each other.

For rows two and four, the resolution system determines that STS should be disabled. This results from the relative priorities of CTS and STS (the former having a higher data collection priority). The choice of priority between these two rules is arbitrary, and would require a domain expert to make this choice in a concrete setting. Further testing with the priorities confirms the goal that inverting the data collection priorities leads to the recommended disabling of CTS. Also setting both data collection priorities to the same value recommends disabling both rules. The results expressed in Table 2 are summarised in Table 3.

Using this form of table, we consider the other examples described in Sections 3.2–3.5. Table 4 shows the results from 4 particular scenarios described earlier, where each scenario involves a particular type of conflict. This table can be expanded to show the results from a number of scenarios as given in Table 5. The paper avoids describing each scenario for space reasons but the table does provide the feature pairs. Although some of the scenarios and individual features have not been described for space purposes, the table shows a set of robust results demonstrating strong detection and resolution capabilities of the technique.

Table 4
Detection and resolution results from four types of conflict.

Conflict type	Rules under test	Detected?	Resolved
SAI	STS & DRF	Yes	Yes
LI	ADT & RDS	Yes	Yes
MAI	DSU & DSTP	Yes	Yes
MTI	Prompt & DNNU	Yes	Yes

Table 5
Expanded list of interaction scenarios.

Conflict type	No.	Rules used	Detect?	Resolve?
STI	1	CTS & STS A	Yes	Yes
SAI	1	RL & Prompt	Yes	Yes
SAI	2	ADT & RDS	Yes	Yes
SAI	3	STS (A and B) & DRF	Yes	Yes
LI	1	ADT & RDS	Yes	Yes
LI	2	RDTOU & RDTOU	Yes	Yes
MAI	1	DSU & DSTP	Yes	Yes
MAI	2	ADT & ODS	Yes	Yes
MAI	3	DNNU & TS	Yes	Yes
MTI	1	DNNU & Prompt	Yes	Yes
MTI	2	Prompt & Prompt	Yes	Yes
MTI	3	CTS & AI	Yes	Yes

7.2. Feature susceptibility

The rules were analysed to determine how prone each feature was to rule conflict. In total, all of the rules presented above were assessed against each other as both the first and second rules in the analysis approach. Out of a total of 867 tests (all of the rules were tested for STI, SAI and MTI), 410 conflicts were detected. Many of the rule combinations may occur rarely (if at all), but it is useful to look at such a wide spread of cases in order to identify patterns that can help developers minimise conflicts.

7.2.1. Feature susceptibility: Shared Trigger Interactions

A high number of shared trigger conflicts were discovered between the rules, so an investigation was used to provide developers with a means of improving concordance. Fig. 6 shows that the rule with the largest number of STI conflicts is the screened form of ODS and the fewest is shared by RL and AI. The range of STI conflicts is from 0 to 23, with a median of 22 and inter-quartile range of 18. The rules are categorised in Table 6 into five categories. The categories were identified as:

1. Those showing high numbers of conflicts in both the first and second rules.
2. Those that have a high number of conflicts in the first rule only.
3. Those that conflict only with themselves and one other.
4. Those that conflict only with themselves.
5. Those that do not conflict.

Ten rules fit into category 1. These rules only conflict with each other when they are the first rules but not with the rules in the other categories. When the rules in category 1 are second rules, however, they conflict with themselves as well as those in category 2. When category 2 rules are first rules, they conflict with all of the rules in category 1 as well as with themselves (but not each other). One of the category 2 rules (TS) also conflicts with a category 3 rule (RDS). Some of the rules such as ODS-Screened are inherently conflict prone. This rule is designed to prevent the operation of other rules for security reasons, so it is understandable that it alters the patterns of behaviour for so many other rules when they share a trigger.

Further analysis was conducted to understand the high number of conflicts involving other rules. An analysis of rule categories 1 and 2 reveals that the high number of conflicts is caused by clauses that alter the states of the triggering message fluents. The difference between the rules in categories 1 and 2 is that those rules in the former category always alters the states of the triggering message fluents, whereas those in the latter category have branching conditions that only alter the states under certain circumstances. The category 5 rule is inert and does not conflict with the other rules as it does not trigger any other activity. It does not contain Σ sentences and therefore the rule cannot be involved with shared triggers. In this case the detection algorithm does not detect shared triggers, and so the behaviour is correct. A naïve response to this might be to develop systems in which the rules never initiate or terminate fluents to avoid shared triggers. A *Data Collection System* (DCS) is an example of such an approach. When the goal, however, is to have automated responses to sensed data, then shared triggers may emerge. To minimise the conflicts seen in categories 3 and 4, where rules only conflict with other instances of themselves, requires constraints on how rules respond to triggers. For example, consider two rules that activate a particular device and share a trigger; only one of them can activate the device because of the binary nature of device activation. Here rule conflict can be stopped if the rules avoid altering their triggering message fluents.

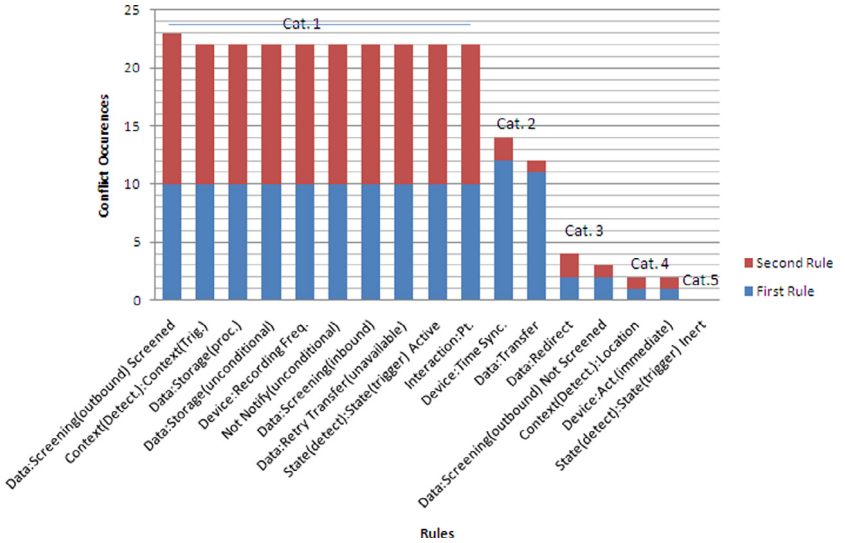

Fig. 6. The number of STI conflicts for 17 rules. The median is 22 and the IqR is 18.

Table 6
STI conflict review.

Category of conflict	Rule	# Conflicts as 1st rule	# Conflicts as 2nd rule	Total # of conflicts
1	ODS screened	10	13	23
1	CTS	10	12	22
1	DSTP	10	12	22
1	DSU	10	12	22
1	DRF	10	12	22
1	DNNU	10	12	22
1	IDS	10	12	22
1	RDTOU	10	12	22
1	STS active	10	12	22
1	Prompt	10	12	22
2	TS	12	2	14
2	ADT	11	1	12
3	RDS	2	2	4
3	ODS not screened	2	1	3
4	RL	1	1	2
4	AI	1	1	2
5	STS inert	0	0	0

7.2.2. Feature susceptibility: Sequential Action Interactions

Fig. 7 shows that ADT has the largest number of SAI conflicts and the unscreened form of ODS has the fewest. The median number of SAI conflicts is 15 with a range between 0 and 24, and an inter-quartile range value of 13. In the study, 8 categories of conflicts are identified for analysis. The seven categories of conflict identified are where the rules have:

1. a medium or high number of conflicts as both the first and second rules
2. a high number of conflicts as the first rules but medium as the second rules
3. a medium number of conflicts as both the first and second rules
4. a high number of conflicts as the first rules but none as the second rules
5. no conflicts as the first rules, but a medium number as the second rules
6. conflicts with only one other rule
7. no conflicts at all.

The categories are used within Table 7 to help summarise the SAI results. The rules within categories 6 and 7 have a lower susceptibility and so provide insights into curtailing SAI conflicts. The category 5 rules prevent the normal functioning of other rules. For instance one conflict is DNNU preventing Prompt from firing as normal. This is a very common pattern for devices such as mobile phones that are put into a silent mode, thereby preventing prompting.

Some AA cases, however, require strict adherence to prompting schedules. Such common behaviour may be overlooked by protocol developers and it is important to tune any priority-based resolution approach through a judicial choice of priorities.

The category 4 rules conflict with ones that terminate a common fluent. The timing of the termination is important. A conflict occurs if a rule terminates the fluent before the category 4 rule checks for branching logic. These rules do not initiate

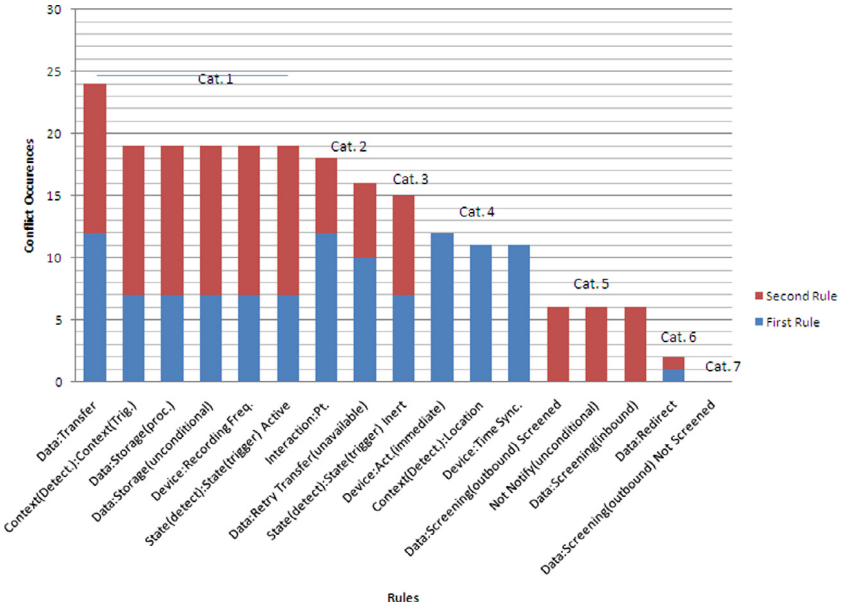

Fig. 7. The number of SAI conflicts for 17 rules. The mean is 13.06. SD is 7.

Table 7
SAI conflict review.

Category of conflict	Rule	# Conflicts as 1st rule	# Conflicts as 2nd rule	Total # of conflicts
1	ADT	12	12	24
1	CTS	7	12	19
1	DSTP	7	12	19
1	DSU	7	12	19
1	DRF	7	12	19
1	STS active	7	12	19
2	Prompt	12	6	18
2	RDTOU	10	6	16
3	STS inert	7	8	15
4	AI	12	0	12
4	RL	11	0	11
4	TS	11	0	0
5	ODS screened	0	6	6
5	DNNU	0	6	6
5	IDS	0	6	6
6	RDS	1	1	2
7	ODS not screened	0	0	0

or terminate any common fluents themselves, therefore they are not the direct cause of a conflict. Minimising actions on common fluents can help reduce these types of conflicts.

The only category 3 rule is STS Inert, and it exposes 8 false positives. Conflicts are reported when there are two instances of this rule because the rule only contained Δ sentences without any branching statements. This is clearly a false positive. It arises because the rule is written to include two instances of the action *listen-for-connection* (the initial and terminal actions) and the SAI conflict rule filters one of them out. When the terminal action is removed from the rule and the conflict check is re-performed, the two instances are in concordance.

7.2.3. Feature susceptibility: Missed Trigger Interactions

Looking at the results of the MTI study in Fig. 8 and Table 8 shows that there are two categories of rules. The first category contains 10 of the rules, and these are conflict prone as both the first and second rules. The other 7 rules are only involved in conflicts as second rules with those of the first category. The median number of MTI conflicts is 27 with a range between 10 and 27, and an inter-quartile range of 17. The 10 rules of category 1 are destructive whereas the others are passive. Destructive rules should be minimised to reduce missed trigger conflicts. Given these forms of the rules, however, the conflict detection algorithm detected results correctly.

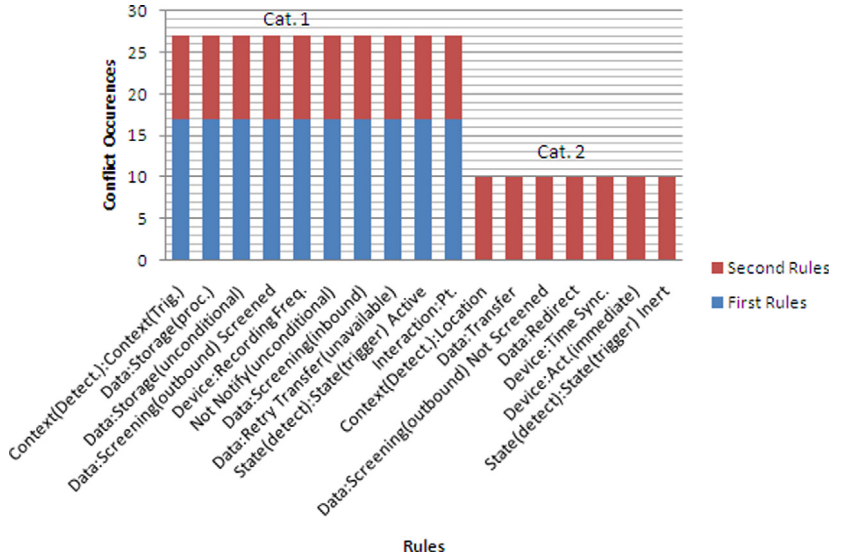

Fig. 8. The number of MTI conflicts for 17 rules.

Table 8
MTI conflict review.

Category of conflict	Rule	# Conflicts as 1st rule	# Conflicts as 2nd rule	Total # of conflicts
1	CTS	17	10	27
1	DSTP	17	10	27
1	DSU	17	10	27
1	ODS screened	17	10	27
1	DRF	17	10	27
1	DNNU	17	10	27
1	IDS	17	10	27
1	RDTOU	17	10	27
1	STS active	17	10	27
1	Prompt	17	10	27
2	RL	0	10	10
2	ADT	0	10	10
2	ODS not screened	0	10	10
2	RDS	0	10	10
2	TS	0	10	10
2	AI	0	10	10
2	STS inert	0	10	10

7.3. Timing analysis

Given the potential of this approach it is important to investigate any performance and scalability constraints. In essence these tests were simply to confirm that the conflict algorithms do indeed have a complexity of $O(n^2)$. They also provide some practical timing values from a typical laptop computer. These tests used the SWI-Prolog *time* statistics feature, and were run on a 2.53 GHz Intel Core 2 Duo CPU with 1.85 GB of RAM. These tests confirm that the conflict algorithms have a complexity of $O(n^2)$. Timing *in the wild* would vary depending on the node that the mediator runs on.

Analysis shows that the conflict detection for the 17 rules used in Section 7.2 above take on average under a tenth of a second for each of the conflict types. For this timing analysis the rule set is doubled a total of six times to give a total of 1088 rules, to produce execution times for larger data sets. For each doubling of the number of rules the analysis time increased by around a factor of four. The analysis times vary depending on the conflict type being searched for, owing to the complexity of the analysis rules. In addition, machine-dependent factors such as other tasks being performed by the CPU in the background influence the timings. To deal with such irregularity, multiple tests were run and average values are reported. Designers can apply a heuristic such that doubling the number of rules will quadruple the time it takes to perform conflict analysis.

Fig. 9 presents results from the timing tests. This shows that MTI detection was quicker than MAI and STI. The increase in the time taken to complete the conflict analysis for double the number of rules ranged between 2.33 and 6.26, with a median value of 4.08.

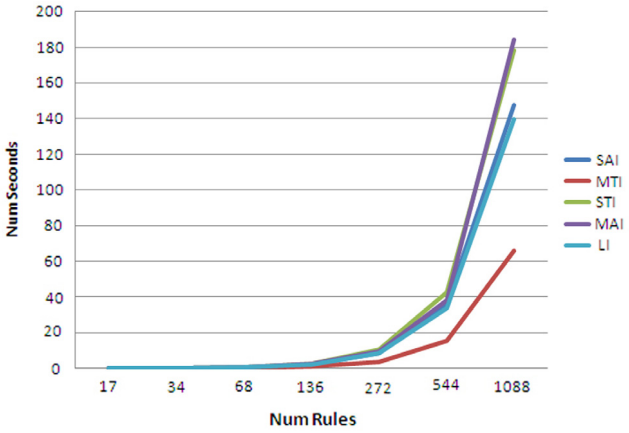

Fig. 9. Timing results chart. This shows that MTI detection was quicker than MAI and STI. There are 6 points on each curve reflecting the 6 doubling of rule numbers. At each point the ratio increase of the time taken to complete the conflict analysis was different. The increase in time ratio for double the number of rules ranged between 2.33 and 6.26, with a median value of 4.08.

8. Future work

In this paper the analysis and resolution are modelled as discrete elements. This separation of concerns provides a useful focus on the nature of the rule conflicts. However for future development of rule-based AA WSNs it is recommended that conflict detection and resolution are more tightly integrated.

The selection of priorities, although driven by experience and the literature, was still essentially subjective in this study. Alternative priorities are conceivable, and such alternatives would be beneficial as part of the personalisation of a future system. However future research on a more subtle resolution approach would be a great assistance.

Future work must address the false positives generated by this approach. Eight false positive results are detected in the SAI analysis, and although these arise in very unusual circumstance, the SAI detection algorithm should be improved in future work to eliminate this. Filtering cases where multiple rules share the same α sentences would be a candidate approach. While no false negatives occurred within this study, there were limits to the number and type of feature examples used, although care was taken in their selection. Also, close study of the SAI and STI algorithms reveals a bias to reporting the worst-case results. In other words they lean towards false positives rather than false negatives. This is exasperated by the binary outcomes of the algorithms. Future work with a larger study would provide more insight here.

Nor can the algorithms determine the quality (goodness or badness) of a particular type of conflict. In other words what impact does a particular conflict have? Some of these conflicts may arise very rarely (if at all) in a given system, and some conflicts may be seen as beneficial. So a more refined outcome from the algorithms would be beneficial.

The MTI algorithm is sensitive to the ordering of events as CLIPPER loses data when it resets its Σ, Δ and Δ_0 predicates prior to analysis. So a problem can arise if a rule needs to be called multiple times before a conflict occurs. Future work could overcome this limitation in at least two ways. Firstly, a recursive call to the given rule can allow it to be called multiple times, allowing it to assert state information that is not reset prior to checks. A second candidate asserts state into the system by composing rules that call the other rules.

Future work could investigate other application areas of this work. Although this paper employs an AA setting the approach it would seem highly likely it could be used in other application areas. The base set of rules may be appropriate for other sensor network monitoring tasks such as environmental monitoring or multi-sensory robotics. Moreover the rule set could be extended to support additional features. For instance, this approach could be used in a call control environment that requires a response to personalised and dynamic rules. Such a situation can potentially arise in a loosely federated system of services across various Voice over IP (VoIP) solutions. For example add a *teenline* feature that restricts outbound calls from a subscriber's phone during particular times of day. To add such a rule, begin by modelling its activities and sequences. Then convert the activity model into EC notation. For the *teenline* example, activities may include events for attempting to connect and checking connection, and there may be fluent for the particular restricted time of day.

Other future work may address extending the error checking beyond rule conflict to include conditions such as resource starvation, race conditions, and deadlocks. The algorithm could be based on firing rules sequentially or in parallel and examining the asserted EC predicates.

In addition to using the detection and resolution approaches highlighted above, rule conflict can be curtailed if rule developers use the insights obtained from the results described above. In particular, when composing rules ensure fluent sharing with other known rules is kept to a minimum and that rules should be kept passive (they should not alter the triggering message) unless there is a particular reason for doing so. The locking mechanism of Wilson [42] is one attempt at managing fluent sharing. Future work to develop a similar approach for the mobile device realm may help to alleviate

some of these difficulties. Some cases, however, will remain when fluents will be shared. If this is the case then it will be important that rule authors are aware of termination timings and take precautions to prevent conflicts that can emerge.

9. Conclusions

In conclusion, it is clear that rule-based Sensor Networks are susceptible to Feature Interaction. There are several problems which can be reduced through careful rule authoring [39] but there are also a large number of intricate interactions which require intelligent network services to detect and resolve. An important aspect of any future rule-based Sensor Networks is that until rules are checked for conflict, the network will remain in an indeterminate state. Thus, it is essential for future network designs and implementations that the rules be checked for interaction before activation and when changes are made to the network.

Acknowledgements

This work was carried with the support of the EPSRC funded project EP/F003684/1 (Enabling health, independence and wellbeing for psychiatric patients through Personalised Ambient Monitoring (PAM)). The PAM project is a collaborative project between the Universities of Stirling, Nottingham and Southampton.

The authors would like to thank all of the PAM research team for their help and support.

References

[1] U. Ebner-Priemer, Society for ambulatory assessment (SAA)—understanding behavior in context, 2010. URL http://www.ambulatory-assessment.org/ (last accessed on 26.05.10).
[2] J. Smyth, A. Stone, Ecological momentary assessment research in behavioral medicine, J. Happiness Stud. 4 (1) (2003) 35–52.
[3] B. Le, H. Choi, D. Beal, Pocket-sized psychology studies: Exploring daily diary software for palm pilots, Behav. Res. Methods 38 (2) (2006) 325.
[4] U. Ebner-Priemer, T. Kubiak, Psychological and psychophysiological ambulatory monitoring: A review of hardware and software solutions, Eur. J. Psychol. Assess. 23 (4) (2007) 214–226.
[5] J.E. Fischer, Experience-sampling tools: a critical review, Mob. Living Labs 9 (2009) 1–3.
[6] L. Barrett, D. Barrett, An introduction to computerized experience sampling in psychology, Soc. Sci. Comput. Rev. 19 (2) (2001) 175–185.
[7] H. Weiss, D. Beal, S. Lucy, S. MacDermid, Constructing EMA Studies with PMAT: The Purdue Momentary Assessment Tool User's Manual, Tech. rep., Purdue University, 2004.
[8] J. Froehlich, M. Chen, S. Consolvo, B. Harrison, J. Landay, MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones, in: Proceedings of the 5th International Conference on Mobile Systems, Applications and Services, ACM, 2007.
[9] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, J. Stankovic, Alarm-net: Wireless Sensor Networks for Assisted-living and Residential Monitoring, Technical Report CS–2006–11, University of Virginia Computer Science Department, 2006.
[10] S. Intille, Technological innovations enabling automatic, context-sensitive ecological momentary assessment, in: The Science of Real-time Data Capture: Self-Reports in Health Research, Oxford University Press, USA, 2007, pp. 308–337.
[11] R. Collins, M. Muraven, Ecological momentary assessment of alcohol consumption, in: The Science of Real-Time Data Capture: Self-Reports in Health Research, Oxford University Press, Inc., 2007, pp. 189–203.
[12] J. Blum, E. Magill, The design and evaluation of personalised ambient mental health monitors, in: Proceedings of the 7th Annual IEEE Consumer Communications and Networking Conference, IEEE, 2010.
[13] J. Blum, E. Magill, The design and evaluation of personalised ambient mental health monitors, in: Consumer Communications and Networking Conference, CCNC, 2010 7th IEEE, 2010, pp. 1–5. http://dx.doi.org/10.1109/CCNC.2010.5421748.
[14] C. James, J. Crowe, E. Magill, S. Brailsford, J. Amor, P. Prociow, J. Blum, S. Mohiuddin, Personalised ambient monitoring (PAM) of the mentally ill, in: J. Vander Sloten, P. Verdonck, M. Nyssen, J. Haueisen (Eds.), 4th European Conference of the International Federation for Medical and Biological Engineering, in: IFMBE Proceedings, Vol. 22, Springer, Berlin, Heidelberg, 2009, pp. 1010–1013.
[15] S. Sen, R. Cardell-Oliver, A rule-based language for programming wireless sensor actuator networks using frequence and communication, in: Proceedings of Third Workshop on Embedded Networked Sensors, EMNETS, 2006.
[16] A. Grunerbl, A. Muaremi, V. Osmani, G. Bahle, S. Ohler, G. Troester, O. Mayora, C. Haring, P. Lukowicz, Smart-phone based recognition of states and state changes in bipolar disorder patients, IEEE J. Biomed. Health Inform. 19 (1) (2014) http://dx.doi.org/10.1109/JBHI.2014.2343154.
[17] K. Terfloth, G. Wittenburg, J. Schiller, Facts—a rule-based middleware architecture for wireless sensor networks, in: Proceedings of the First International Conference on Communication System software and Middleware, COMSWARE'06, New Delhi, India, 2006.
URL: http://page.mi.fu-berlin.de/~terfloth/terfloth06facts.pdf.
[18] X. Fei, E. Magill, REED: Flexible rule based programming of wireless sensor networks at runtime, Comput. Netw. 56 (14) (2012) 3287–3299.
[19] M. Negnevitsky, Artificial Intelligence a Guide to Intelligent Systems, Pearson Education Limited, 2002.
[20] A.J. Gonzalez, D.D. Dankel, The Engineering of Knowledge-Based Systems Theory and Practice, Prentice-Hall, Inc., 1993.
[21] P. Jackson, Introduction to Expert Systems, third ed., Addison Wesley Longman Limited, 1999.
[22] T. Bowen, F. Dworack, C. Chow, N. Griffeth, G. Herman, Y. Lin, The Feature Interaction problem in telecommunications systems, in: Seventh International Conference on Software Engineering for Telecommunication Switching Systems, 1989. SETSS 89, 1989, pp. 59–62.
[23] R. Crespo, M. Carvalho, L. Logrippo, Distributed resolution of Feature Interactions for Internet applications, Comput. Netw. 51 (2) (2007) 382–397.
[24] M. Wilson, M. Kolberg, E. Magill, Considering side effects in service interactions in home automation-an online approach, Feature Interact. Softw. Commun. Syst. 9 (2007) 172–187.
[25] M. Nakamura, H. Igaki, Y. Yoshimura, K. Ikegami, Considering online Feature Interaction detection and resolution for integrated services in home network system, in: 10th International Conference on Feature Interactions in Telecommunications and Software Systems, ICFI2009, 2009, pp. 191–206.
[26] C. Maternaghan, K.J. Turner, Policy conflicts in home automation, Comput. Netw. 57 (12) (2013) 2429–2441.
[27] M. Nakamura, K. Ikegami, S. Matsumoto, Considering impacts and requirements for better understanding of environment interactions in home network services, Comput. Netw. 57 (12) (2013) 2442–2453.
[28] M. Kolberg, J.F. Buford, K. Dhara, X. Wu, V. Krishnaswamy, Feature interaction in a federated communications-enabled collaboration platform, Comput. Netw. 57 (12) (2013) 2410–2428.
[29] J. Xu, K. Chen, Y. Duan, S. Reiff-Marganiec, Modeling business process of web services with an extended strips operations to detection Feature Interaction problems runtime, in: 2011 IEEE International Conference on Web Services, ICWS, IEEE, 2011, pp. 516–523.
[30] S. Kolesnikov, J. Roth, S. Apel, On the relation between internal and external Feature Interactions in feature-oriented product lines: a case study, in: Proceedings of the 6th International Workshop on Feature-Oriented Software Development, ACM, 2014, pp. 1–8.

[31] M. Calder, M. Kolberg, E. Magill, S. Reiff-Marganiec, Feature interaction: a critical review and considered forecast, Comput. Netw. 41 (1) (2003) 115–141.
[32] D. Marples, Detection and resolution of Feature Interactions in telecommunications systems during runtime (Ph.D. thesis), University of Strathclyde, 2000.
[33] K.J. Turner, G.A. Campbell, Goals for telecare networks, in: A. Obaid (Ed.), Proc. 9th Int. Conf. on New Technologies for Distributed Systems, 2009, pp. 270–275.
[34] A. Bandara, E. Lupu, A. Russo, Using event calculus to formalise policy specification and analysis, in: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks, IEEE Computer Society, 2003, pp. 26–39.
[35] J. Park, M. Moon, S. Hwang, K. Yeom, CASS: a context-aware simulation system for smart home, in: 5th ACIS International Conference on Software Engineering Research, Management & Applications, 2007, SERA 2007, IEEE, 2007, pp. 461–467.
[36] T. Zhang, B. Brügge, Empowering the user to build smart home applications, in: ICOST 2004 International Conference on Smart Home and Health Telematics, 2004.
[37] J.-Y. Jung, J. Park, S.-K. Han, K. Lee, An ECA-based framework for decentralized coordination of ubiquitous web services, Inf. Softw. Technol. 49 (11) (2007) 1141–1161.
[38] P. Carreira, S. Resendes, A.C. Santos, Towards automatic conflict detection in home and building automation systems, Pervasive Mob. Comput. 12 (2014) 37–57.
[39] J. Blum, Handling emergent conflicts in adaptable rule-based sensor networks (Ph.D. thesis), Computing Science and Mathematics, School of Natural Sciences, University of Stirling, 2012.
[40] R.J. Larsen, Personality, mood states, and daily health, in: The Science of Real-Time Data Capture: Self-Reports in Health Research, Oxford University Press, Inc., 2007, pp. 251–267. (Chapter).
[41] M. Calder, P. Gray, C. Unsworth, Tightly coupled verification of pervasive systems, Electron. Commun. EASST 22 (2009) 1–16. (Open Access).
[42] M.E.J. Wilson, An online environmental approach to service interaction management in home automation (Ph.D. thesis), University of Stirling, 2005.
[43] M. Kolberg, Service interaction management in a deregulated market environment (Ph.D. thesis), University of Strathclyde, 2004.
[44] R. Miller, M. Shanahan, The event calculus in classical logic—alternative axiomatisations, Comput. Inf. Sci. 4 (1999) 16.
[45] R.M.D. Reiff-Marganiec Stephan (Ed.), Handling Policy Conflicts in Call Control, IOS Press, 2005.