A Framework for the Design and Evaluation of Network Resilience Management

Alberto Schaeffer-Filho, Paul Smith, Andreas Mauthe and David Hutchison
School of Computing and Communications,
InfoLab21, Lancaster University, UK
Email: {asf, p.smith, andreas, dh}@comp.lancs.ac.uk

Yue Yu and Michael Fry
School of Information Technologies,
University of Sydney,
NSW 2006, Australia
Email: {tinayu, mike}@it.usyd.edu.au

***Abstract*—Network resilience strategies aim to maintain acceptable levels of network operation in the face of challenges, such as malicious attacks, operational overload or equipment failures. Often the nature of these challenges requires resilience strategies comprising mechanisms across multiple protocol layers and in disparate locations of the network. In this paper, we address the problem of resilience management and advocate that a new approach is needed for the design and evaluation of resilience strategies. To support the realisation of this approach we propose a framework that enables (1) the offline evaluation of resilience strategies to combat several types of challenges, (2) the generalisation of successful solutions into reusable patterns of mechanisms, and (3) the rapid deployment of appropriate patterns when challenges are observed at run-time. The evaluation platform permits the simulation of a range of challenge scenarios and the resilience strategies used to combat these challenges. Strategies that can successfully address a particular type of challenge can be promoted to become resilience patterns. Patterns can thus be used to rapidly deploy resilience configurations of mechanisms when similar challenges are detected in the live network.**

I. Introduction

Computer and communication networks are increasingly critical in supporting business, leisure and daily life in general. There is also an evident increase in cyber attacks on networked systems. Thus, there is a compelling need for *resilience* to be a key property of networks. Resilience is the ability of the network to maintain acceptable levels of operation in the face of challenges, such as malicious attacks, operational overload, mis-configurations, or equipment failures [1]. This paper describes a framework and a process for providing *resilience management*, which encompasses and perhaps supplants some elements of the traditional FCAPS (fault, configuration, accounting, performance, and security) functionalities. The nature of the challenges typically requires the use of mechanisms across multiple layers of the protocol stack and in disparate locations of the network. Therefore, ensuring the resilience of a network requires the systematic design and evaluation of resilience strategies, the careful coordination of various monitoring and control mechanisms, and also the capture of best practices and the experience of network operators into reusable resilience configurations.

Network resilience is considered in the context of a general two-phase high-level strategy, called $D^2R^2 + DR$: *Defend*, *Detect*, *Remediate*, *Recover* + *Diagnose*, *Refine* [1]. The first phase comprises the use of *defensive* measures to protect the network from foreseeable challenges, the ability to *detect* in real-time challenges that have not been anticipated and subsequently *remediate* their effects before the network operation is compromised, and finally disengage possibly sub-optimal mechanisms via specific *recovery* procedures. The second phase caters for the longer-term evolution of the system, through the *diagnosis* of the causes of the challenge and the *refinement* of the system operation. In particular, D^2R^2 can be seen as a conceptual online control-loop for network resilience operation. Central to the strategy is the management and reconfiguration of interacting detection and remediation mechanisms operating in the network infrastructure.

On the one hand, detection mechanisms such as link monitors, anomaly detection systems and traffic classifiers permit the identification and categorisation of challenges to the network. On the other hand, remediation mechanisms such as rate limiters and firewalls are used in the subsequent mitigation of these challenges. Recently, we have proposed the notion of *multi-stage resilience strategies* [2], in which the configuration of detection and remediation mechanisms deployed in the network is dynamically refined as new information about challenges becomes available. Policies are used to control the operation of these mechanisms, and how they should be reconfigured in the face of new types of challenges or changes in their operational context.

In such resilience strategies, network components implementing a range of resilience functions must be autonomous and capable of continuously adapting their operation to, for example, high resource utilisation, performance degradation, or application-specific alarms. Therefore, we assume that each resilience mechanism is capable of implementing a policy-driven feedback control-loop, such as the one provided by the *Self-Managed Cell* (SMC) [3] infrastructure.

This paper presents an integrated framework for the design and evaluation of resilience strategies. Through an online *challenge analysis* approach, it is possible to collect and correlate network metrics and traffic information, in order to build an up-to-date understanding of possible challenges

978-1-4673-0269-2/12/$31.00 © 2012 IEEE

affecting the network. Resilience strategies for addressing a specific challenge are specified as policy-driven configurations between a range of resilience mechanisms in the *network infrastructure*. Such resilience strategies are likely to span multiple autonomous domains and protocol layers, and for these reasons they need to be established systematically. To achieve this, we define *reusable patterns* [4] for building policy-based configurations that can be dynamically established between sets of resilience mechanisms. Finally, we provide support for the offline evaluation of these policy-driven resilience strategies through a *simulation environment*. This permits the simulation of network challenges (e.g., DDoS attacks, flash crowds, worm propagations) and the evaluation of the performance of a set of policy-driven resilience mechanisms.

The remainder os the paper is structured as follows: Section II describes a process for the design and evaluation of network resilience, and Section III presents the overall framework that supports this process. Section IV outlines the specification of patterns for network resilience. Section V discusses how patterns can be evaluated offline using a simulation environment. Finally, Section VI presents the related work and Section VII outlines the concluding remarks.

II. A Network Resilience Process

This section defines a *process* for the design and evaluation of strategies for network resilience. The basic idea is that one must be able to *(1)* perform an offline evaluation of resilience strategies to combat specific types of challenges, then *(2)* generalise successful solutions into reusable patterns of resilience mechanisms, and finally be able to *(3)* select and deploy appropriate patterns to address these challenges when they are observed during run-time.

The evaluation of large-scale challenges, such as DDoS attacks, is difficult as resilience strategies for such challenges often require the coordination of various monitoring and control mechanisms in different parts of the network. The use of testbeds can involve high costs of hardware and development effort, and are generally not suitable for the evaluation of large-scale challenges, which tend to affect multiple autonomous systems. As an alternative, to mitigate costs and address scaling issues, we advocate the reproduction of network challenges and resilience mechanisms in a simulation environment.

In our work, resilience strategies are expressed using policies. We have developed a *policy-driven resilience simulator*, which is used to evaluate resilience strategies [5]. Strategies that perform well in the simulation environment are promoted to become resilience patterns. Patterns encode the overall configuration of a set of resilience mechanisms, but will not determine the exact instances or the precise parameterisation of these mechanisms. Instead, these are determined at the point of deployment, based on the current availability and capability of the devices associated with a specific network. Patterns can be used to rapidly reconfigure devices in the infrastructure when challenges manifest in the live network.

The process we propose is illustrated in Fig. 1, and its three key parts are described in the following:

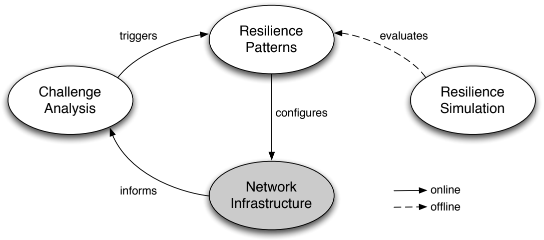

Fig. 1. Process for the design & evaluation of network resilience.

Challenge analysis: relies on an online monitoring infrastructure to gather and store information about the current state of the network. Metrics can be correlated to produce higher level information, and trigger the reconfiguration of the network.

Resilience simulation: permits the simulation of different challenge scenarios and the evaluation of policy-based configurations of resilience mechanisms to combat these challenges.

Resilience patterns: resilience configurations that perform well against specific challenges in the simulation environment are promoted to *reusable patterns* [4]. Patterns specify a policy-driven configuration between a set of abstract mechanisms types.

The simulation environment is valuable for the understanding of the different challenge profiles and candidate mitigation strategies. By capitalising on successful resilience configurations, one can derive generalised patterns for coping with different challenge behaviours. Patterns thus support the notion of reusing tested solutions for well-known problems when building strategies for network resilience.

III. Policy-driven Resilience Framework Overview

In order to support the realisation of the process discussed previously, we designed a resilience framework, which is illustrated in Fig. 2. At its core is a *resilience manager*, which is tasked with orchestrating network adaptation to ensure the resilience of the network and its supported services. Adaptation of network operation can occur in response to either the detection of the onset of a challenge or the measured performance of the network in relation to resilience targets, expressed in Service Level Agreements (SLAs). These two factors lead to a set of *coarse-grain* and *fine-grain* adaptations.

Coarse-grain adaptation involves the deployment of resilience patterns, which are configurations of resilience mechanisms, capable of combating a specific challenge. For this, a *challenge analysis* module provides information about the challenges that are affecting a network. Fine-grain adaptation involves setting or adjusting the parameters of the mechanisms that are currently deployed in the network as part of an existing pattern. In this case, the *resilience manager* evaluates to what extent a resilience target is being met, causing this fine-grain adaptation. Both sets of adaptations, i.e., coarse-grain pattern deployment and fine-grain mechanism configuration, are expressed in terms of event-triggered condition-action (ECA) policies, enforced by the resilience manager.

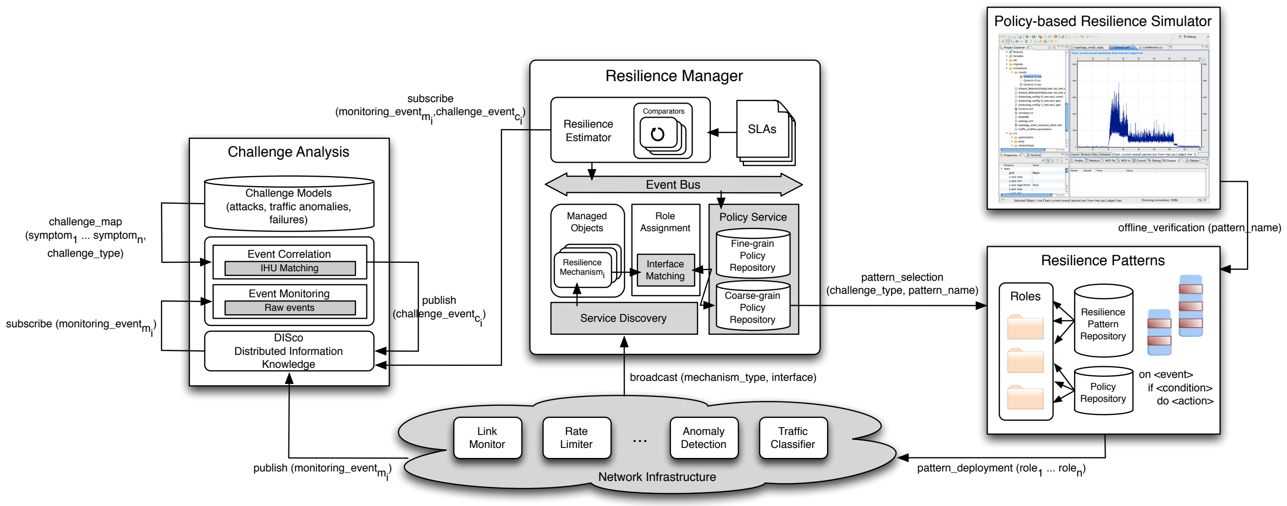
Policy-based Resilience Simulator
Resilience Manager
Resilience Estimator
Comparators
SLAs
Event Bus
Managed Objects
Resilience Mechanism_i
Role Assignment
Interface Matching
Policy Service
Fine-grain Policy Repository
Coarse-grain Policy Repository
Service Discovery
Challenge Analysis
Challenge Models (attacks, traffic anomalies, failures)
Event Correlation
IHU Matching
Event Monitoring
Raw events
DISco Distributed Information Knowledge
challenge_map (symptom_1 ... symptom_n, challenge_type)
subscribe (monitoring_event_mi)
subscribe (monitoring_event_mi, challenge_event_ci)
publish (challenge_event_ci)
pattern_selection (challenge_type, pattern_name)
offline_verification (pattern_name)
Resilience Patterns
Roles
Resilience Pattern Repository
Policy Repository
on <event> if <condition> do <action>
broadcast (mechanism_type, interface)
Link Monitor
Rate Limiter
...
Anomaly Detection
Traffic Classifier
Network Infrastructure
publish (monitoring_event_mi)
pattern_deployment (role_1 ... role_n)

Fig. 2. Overview of the framework for specification and evaluation of network resilience strategies.

In this section we present the overall design of the resilience manager and the challenge analysis module. The specification of resilience patterns and the validation of patterns using the simulation environment will be discussed in detail in sections IV and V, respectively.

A. Resilience Manager

The resilience manager runs a policy-driven feedback control-loop, and its implementation is based on the *Self-Managed Cell* (SMC). It relies on a set of management services, including: a *policy service*, which supports both obligation and authorisation policies; an *event service* that carries events between other components and the policy service; and a *discovery service* capable of discovering available resources, as we assume the existence of a number of mechanisms implementing a range of resilience functions and services in the network. Such discovered mechanisms are catalogued as *managed objects* by the resilience manager, and will be later assigned to one or more roles in a pattern, according to the functionality supported by each mechanism. The resilience manager resides in a single autonomous domain, and coordinates the transitions between the different stages of the resilience process described in Section II, based on the challenges observed and mechanisms available.

The *resilience estimator* service, shown in Fig. 3, is an extension of the core set of management services supported by the resilience manager. It subscribes to events generated by the challenge analysis module, and its purpose is to determine whether a desired resilience target, defined in an SLA, is being met. An SLA describes the target behaviour of a set of metrics $\{\mathrm{m}_0 \ldots \mathrm{m}_{n-1}\}$ at various levels of the protocol stack, e.g., from average node degree to end-to-end delay. For each metric m_i there is a corresponding *metric comparator* that takes measurement information collected from distributed monitoring mechanisms and the behaviour described in the SLA for m_i, and determines whether a *metric_change_event* should be generated.

Sterbenz *et al.* [1] propose a framework for multi-level resilience metrics that can be used to determine boundaries for acceptable, degraded and unacceptable metric performance; a comparator can use transitions between these states to generate metric change events. Furthermore, temporal and spatial constraints could be described, e.g., generate an event after one minute of degraded performance for a given subnetwork. Such events are used to trigger fine-grain ECA policies, such as adjustments to the parameters and thresholds of resilience mechanisms currently deployed[1].

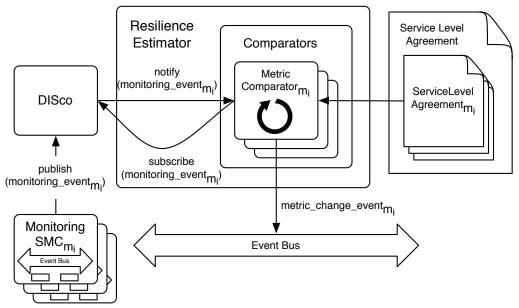

Fig. 3. The resilience estimator.

B. Challenge Analysis

The aim of the challenge analysis component is to develop *situational awareness* [6] with respect to the existence and nature of challenges. It uses events published by the monitoring infrastructure to *perceive* symptoms of a challenge. Monitoring information is stored in *DISco* [7], which provides a common information dispatching and persistence service. This information is subsequently correlated to enable *comprehension*, i.e., identification, of a challenge. There are a number of techniques for identifying a challenge, including data fusion [8]; we propose to build on an approach by Steinder and Sethi, called Incremental Hypothesis Updating (IHU) [9].

[1]In some circumstances, however, the generation of metric change events, e.g., caused by a transition from acceptable to impaired metric state, could also lead to coarse-grain adaptation.

IHU is a probabilistic fault localisation technique that can be used to identify a set of faults that are the cause of symptoms observed by various monitoring functionalities, e.g., as a consequence of SNMP traps. Faults f and their symptoms s are modelled as a symptom-fault map, using a bipartite directed graph. (We generalise the use of symptom-fault maps to describe relationships between challenges, in addition to faults, and their symptoms.) In a *challenge_map*, the set of challenges to be modelled is connected to a set of symptoms that are observed if the challenge manifests. The IHU technique, on receipt of a newly observed symptom, yields a set hypotheses explaining the observed symptoms to-date. Furthermore, a measure of relative *belief* in the correctness of each hypothesis is given. Challenges described in the hypothesis with the highest belief measure will cause a *challenge_event* to be published into *DISco*, and subsequently relayed to the *resilience manager*.

Based on the *challenge_event* observed, *pattern_selection* is performed by the resilience manager via its coarse-grain ECA policies. At *pattern_deployment*, mechanism instances providing the required functionality for each role defined in a pattern are assigned by the resilience manager. According to its role, each mechanism will have policies and event-handling code transferred to it in order to enforce the pattern's semantics. We further discuss the specification and use of resilience patterns in the next section.

IV. Reusable Patterns for Resilience Strategies

To assist the building of federated policy-driven mechanisms implementing a resilience strategy, we rely on the notion of *reusable patterns* [4]. In this paper, a pattern is a policy configuration of resilience mechanisms and their relationships. Patterns are used to address a particular type of network challenge. Different challenge types will demand specific sets of mechanisms to monitor features in the network (e.g., current traffic load or alarms generated by an anomaly detection mechanism), and initiate remediation actions to combat anomalous behaviour (e.g., blocking malicious flows or selectively dropping packets). This assumes the existence of autonomous mechanisms supporting a range of resilience functions in the network, implemented as policy-enabled SMCs.

A. Pattern Specification

Patterns are abstractly specified in terms of *roles*, to which management functions and policies are associated with. Roles represent mechanism types. At instantiation, a pattern will have mechanism instances (i.e., SMCs) assigned to its roles. Patterns can use different, but *compatible*[2], sets of mechanisms available in the network, thus avoiding the need to know during specification time the exact instances that will be used.

A pattern specification (Fig. 4) consists of:

[2]This assumes that the SMCs are *compatible* with the roles, which means that an SMC must provide an interface that supports the events and policies required by the role. In Fig. 4, it is expected that $Role_n$ will be performed by an SMC that supports the operation *classifyFlows* and that $Role_2$ will be performed by an SMC that supports the operation *recordFlows*, for example.

(a) the types of required mechanisms (represented by *roles*)
(b) how these mechanisms must interact with each other.

For example, a pattern for combating a flash crowd challenge may include roles such as *VMReplicator* and *WebServerMonitor*, whereas a pattern for addressing a DDoS attack may include roles such as *TrafficClassifier* and *RateLimiter*. Note, the same mechanism instance may be assigned to more than one role. For example, an enhanced router [2] may perform the generation of netflow records and also filter off determined malicious flows.

In addition to roles, a pattern also defines their management relationships, in terms of the policies that should be loaded and the events that should be exchanged. These relationships are expressed in terms of more primitive interactions (*styles*), which are used as building blocks to connect the roles in the pattern. A catalogue of styles defining common management relationships, such as *p2p*, *event diffusion*, *hierarchical management*, has been presented in [4]. Each style defines its own style-specific roles (e.g., *source/target*, for *event diffusion*). The manner in which style-specific roles are connected establishes the relationships between the roles in the pattern.

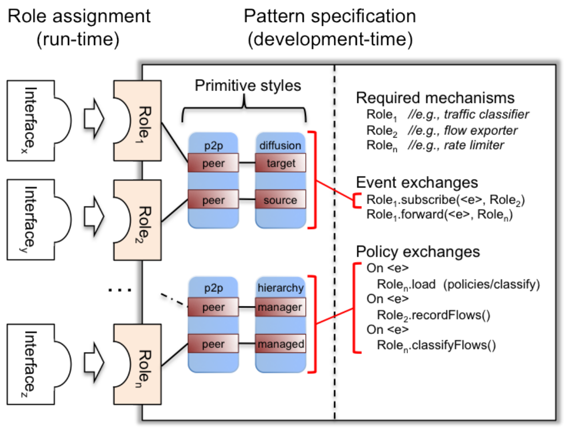

Fig. 4. Pattern specification and role assignment.

B. Pattern Deployment

We assume that mechanism instances providing a range of management functions are initially catalogued by the resilience manager (Fig. 2). When the onset of a challenge is identified by the *challenge analysis* module, *coarse-grain ECA policies* determine which pattern(s) must be deployed. Roles in the chosen pattern prescribe the mechanism types required to combat that challenge. To these roles, specific mechanism instances are assigned.

Role assignment is defined in terms of a set of *events (E)* that must be sent, *notifications (N)* that must be handled, and *operations (O)* that must be supported by an SMC [4]. Formally, let $Itf_m = \langle O_m, E_m, N_m \rangle$ be the management interface of an SMC, let r be a role, and let $Req_r = \langle O_r, E_r, N_r \rangle$ be the requirements for that role, then the assignment of the SMC which has Itf_m to role r is subject to the following:

$$assign(Itf_m, r) \rightarrow (O_r \subseteq O_m) \wedge (E_r \subseteq E_m) \wedge (N_r \subseteq N_m)$$

TABLE I
ROLES CONTAINED IN A PATTERN TO COMBAT HIGH-VOLUME TRAFFIC CHALLENGES, SUCH AS A DDOS ATTACK.

AnomalyDetection	Performs online analysis to identify anomalies in network traffic, for example, anomalous changes in the entropy distribution of ports, IP addresses, or other statistical features
IPFlowExporter	Responsible for generating IP flow records using a specific sampling rate, which are truncated and sent at specific time out period, e.g., 60s or 180s, to other mechanisms to process the records
RateLimiter	Selectively throttles network traffic in specific links when anomalous activity is detected but not yet determined (suspicious but not necessarily malicious)
TrafficClassifier	Receives IP flow records and applies machine learning algorithms to identify the precise nature of the flow information, e.g., a flow is benign or part of a TCP SYN attack

A given pattern may require roles with specific capabilities in terms of the ability to, e.g., monitor links, capture flows or classify traffic. Role assignment provides a way of *type-checking* a mechanism instance in order to verify its suitability for a given role. A pattern defines the overall configuration of mechanisms, but the parameters and thresholds of these mechanisms are defined according to *fine-grain ECA policies*, which take as input the most recent *metric_change_event* published by the resilience estimator (Fig. 3).

C. Example Scenario: High-Volume Traffic Challenge

Fig. 5 illustrates part of the *textual specification*[3] of a pattern to combat high-volume traffic challenges, e.g., a DDoS attack. The pattern is parameterised with four roles, which are described in Table I. The relationships between the roles are defined in terms of the policies that must be loaded and the events that must be exchanged. In lines 6-14, styles are used to establish these relationships. In particular, we are setting an event diffusion between `IPFlowExporter` and `Classifier` (lines 6-8), another event diffusion between `AnomalyDetection` and `RateLimiter` (lines 9-11), and a hierarchical policy loading between `AnomalyDetection` and `RateLimiter` (line 12-14).

A set of policies that can be loaded into another system is called a *mission* [3]. Fig. 5 defines two missions, `throttling` in lines 16-19 (to be loaded into `RateLimiter`) and `detection` in lines 21-30 (to be loaded into `AnomalyDetection`). The former specifies a simple rate limiting policy when a specific IP address is deemed suspicious, and the latter defines what should occur when an anomaly is detected with a certain confidence level (*%y*), in particular flag the anomaly, configure `Classifier` to use a specific algorithm, and generate an alarm.

A pattern is instantiated only if the available SMC instances satisfy the requirements for their roles, as discussed in Section IV-B. This ensures that the policies inside the pattern can be executed by these SMCs. Patterns facilitate the systematic building of policy-driven configurations, and can

[3]We use a succinct pseudo syntax but in the current implementation patterns are written in *PonderTalk [10]* which is more verbose. We also limit the example to the configuration of a small set of mechanisms.

```
 1 type pattern HighVolumeTraffic (role AnomalyDetection,
 2                                 role IPFlowExporter,
 3                                 role RateLimiter,
 4                                 role Classifier)
 5 {
 6   bind style diffusion (target Classifier,
 7                         source IPFlowExporter)
 8                         event: notify_new_record(flow);
 9   bind style diffusion (target RateLimiter,
10                         source AnomalyDetection)
11                         event: notify_detection(IPAddress);
12   bind style hierarchical (manager AnomalyDetection,
13                         managed RateLimiter)
14                         mission: throttling;
15
16   mission throttling {          //loaded into RateLimiter
17     on notify_detection(IPAddress)
18       do limit(IPAddress, %x);
19   }
20
21   mission detection {       //loaded into AnomalyDetection
22     on notify_load(name, rate, link)
23       if ((process(link, IPAddress) > %y) &&
24              anomalyList notContain(IPAddress))
25         do {
26           anomalyList add(link, IPAddress));
27           Classifier setAlgorithm(KNearestNeighbors, %z);
28           generateAlarm notify_detection(IPAddress);
29         }
30     }
31 }
```

Fig. 5. The resilience pattern defines the overall configuration of mechanisms, but the parameters and thresholds (*%x, %y and %z above*) are defined according to the most up-to-date metrics published by the resilience estimator.

also be reused to cater for similar challenges that manifest at different parts of the network, or variations of an attack.

V. RESILIENCE SIMULATION

To evaluate the performance of resilience strategies, we have developed a *policy-based resilience simulator* [5]. The toolset supports the simulation of a range of network challenges, as well as the reproduction of the policy-driven interactions between the mechanisms used to combat such challenges. Resilience strategies for a particular challenge might use different combinations of mechanisms and policies, and the strategies that perform well in the simulation environment can be subsequently promoted to reusable patterns.

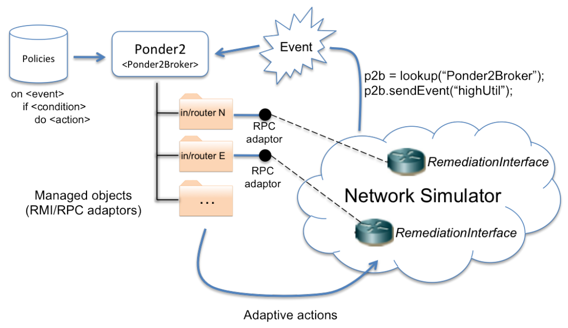

Fig. 6. Policy-based resilience simulator implementation [5].

A. Simulation Platform

The toolset is based on an integration between the OMNeT++ network simulator [11] and the Ponder2 policy framework [10]. It allows the use of policies to define which mechanisms must be activated on demand according to events observed in the simulated network (as opposed to hardcoded

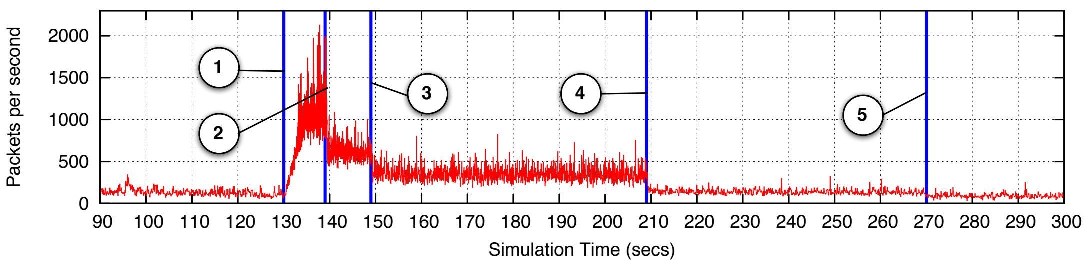

Fig. 7. Simulation results for the DDoS resilience strategy [2].

protocols). An event broker resolves event notifications in the simulation (e.g., anomaly detections, link load) to the policy framework. Fig. 6 presents an overview of the resilience simulator implementation[4].

Resilience mechanisms are implemented as instrumented components that run alongside standard simulated objects. For the challenge scenarios discussed in this section, we have implemented a set of resilience mechanisms, most as extensions of OMNeT++'s standard `Router` module. `FlowExporter` and `AnomalyDetection` are positioned above the network layer, and receive duplicate packets. `RateLimiter` is placed in-line between the network and physical layers. Finally, `LinkMonitor` was implemented by modifying an existing channel type, and can be placed at any position of the topology. Each instrumented object defines a management interface specifying which operations it supports. Management interfaces are used by Ponder2 for the invocation of operations on the simulated objects. Communication between Ponder2 and OMNeT++ is implemented using XMLRPC[5].

B. Challenge Simulation

To illustrate our approach we summarise the results of two challenge scenarios: a *DDoS attack* and a *worm propagation*. We use *ReaSE*[6] to create realistic topologies and generate background and attack traffic. In particular *ReaSE* can generate DDoS attack traffic based on the Tribe Flood Network [12] and also attack traffic that simulates Code Red Worm propagation. Further details of the simulation set-up can be found in [2].

For the DDoS challenge scenario, we simulated a network consisting of twenty Autonomous Systems (ASes): fourteen stub ASes connected by six transit ASes. A Web server in one of the stub ASes is configured as the victim to be attacked by thirty nine *DDoSZombie* hosts across the network. In addition, 1105 hosts generate background traffic to a number of other servers in the network. The resilience strategy evaluated (shown in Fig. 8) consists of various mechanisms activated on the ingress link from a core router to the gateway of the AS under attack. That is, the functions of monitoring, anomaly detection, rate limiting and so on are carried out at the edge of the AS network, in order to protect the AS's network.

Fig. 7 shows the observed incoming traffic rate on the ingress link of the autonomous system under attack. At

[4]We have set up a website with instructions for downloading, installing and executing the simulator: http://www.comp.lancs.ac.uk/resilience/software.

[5]http://xmlrpc-c.sourceforge.net/

[6]https://i72projekte.tm.uka.de/trac/ReaSE/wiki

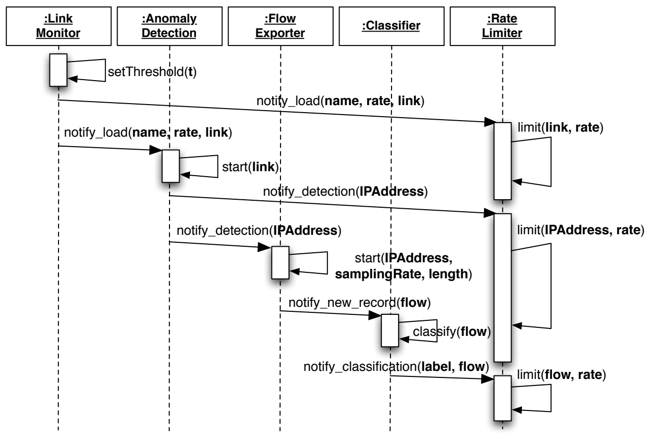

Fig. 8. DDoS resilience strategy using five types of mechanisms.

approximately 130 seconds (1), the onset of the attack can be observed. The raising of an alarm (`notify_load`) by `LinkMonitor` is seen at 139 seconds (2), whereby a sustained traffic load in excess of the threshold defined in the policies has been reached (here, an increase in average incoming traffic of four times the previous average). Shortly thereafter, initial rate limiting of the ingress link by `RateLimiter` can be observed. The filtering rate is likewise set by a policy. In this case we filter 70% of all incoming traffic in order to protect downstream servers and infrastructure. Results show that 92% of blocked traffic during this period is malicious.

At 149 seconds (3), `AnomalyDetection` identifies the destination IP address of the victim. This is achieved by examining the destination address of each incoming packet, and raising an event (`notify_detection`) when one destination accounts for 60% of all packets. `RateLimiter` is then reconfigured to drop 70% of the traffic destined for the victim only. Legitimate traffic that is not destined for the victim, which previously was blocked, is now allowed to go through. Results show that in this period, 95% of blocked traffic is malicious, while the proportion of legitimate traffic that is not blocked increases compared to the previous period.

`Classifier` is initiated at (3) and flow exporting from the router is started. Hereafter, `Classifier`, receiving flow records from the `FlowExporter`, attempts to identify the specific attack flows. At 209 seconds (4), an event with the classification results is generated (`notify_classification`), rate limiting is confined just to the attack flow and legitimate traffic can continue. Results show that after (4) no legitimate traffic is blocked. In this

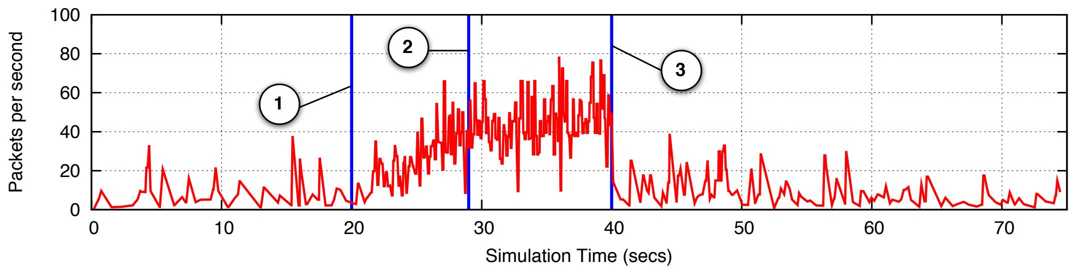

Fig. 9. Simulation results for the Worm resilience strategy.

scenario, the strategy consists of using lightweight detection initially, and then progressively applying more heavyweight analysis to identify the specific challenge. In parallel, it applies coarse-grain remediation initially, to maintain a level of normal operation, which then moves to a more fine-grain remediation.

We evaluated a further strategy to address a worm propagation. We simulated the same AS network topology as above. Hosts are randomly chosen to undertake Code Red worm propagation across the network. For our simulated scenario a maximum of 5000 probing packets are sent. In addition, 930 hosts, 71 web servers and 21 interactive servers generate background traffic across the network. As above, the various mechanisms are activated on the ingress link from a core router to the gateway of an AS. The resilience strategy evaluated (shown in Fig. 10) uses `LinkMonitor` and `RateLimiter` from the DDoS scenario, but reconfigures them using different policies to implement a different resilience strategy

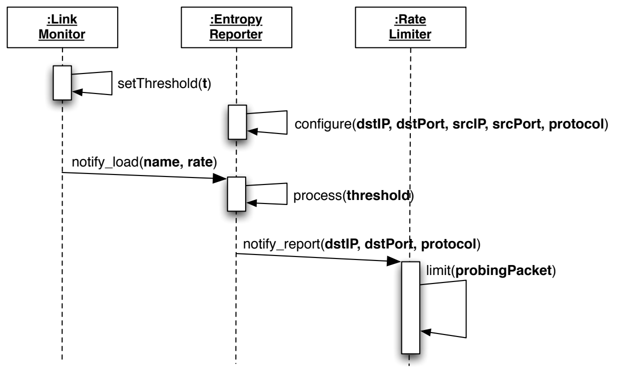

Fig. 10. Worm resilience strategy using three types of mechanisms.

At the start of the simulation, `LinkMonitor` is activated with an alarm threshold set to an increase in average incoming traffic of twice the previous average (i.e., lower than the threshold set in the previous scenario). `EntropyReporter` is also activated to collect periodically (every ten seconds) packet-level entropy values on five traffic features: destination IP, destination port, source IP, source port and protocol.

Fig. 9 shows the worm propagation starting at approximately 20 seconds (1). This increases the volume of traffic on the ingress link. An alarm (`notify_load`) is raised by `LinkMonitor` at 29 seconds (2). Following this event, `EntropyReporter` is interrogated for any significant changes in the five traffic features, as specified in a threshold parameter. The most recent entropy trend for each feature is computed, with results showing that destination IP has become dispersed, destination port centralised, and UDP protocol more dispersed. These results are reported back (`notify_report`) at 40 seconds (3). The resilience strategy implemented by policies recognises that these changes in entropy signify a Code Red worm. Consequently, at (3) `RateLimiter` is configured to filter all probing packets, specified as all UDP packets with a destination port 80. These experiments provide evidence of how specific policy-driven strategies using different sets of resilience mechanisms can be evaluated in the simulation environment, and show how challenges (for example, a flooding attack or a worm propagation) can be dealt with. For different resilience strategies, we are able to use different mechanism types and reconfigure their operation via different policies.

VI. Related Work

The framework presented in this paper builds on parts of our previous efforts on network resilience [1], [2]. It also employs research done independently by the authors. In particular, Self-Managed Cells are used to provide the implementation of a policy-driven feedback control-loop. SMCs have been used in several application domains in the past, including in the management of unmanned autonomous vehicles (UAVs) [13] and healthcare applications [3]. Although the strategies for network resilience discussed in this paper use the SMC model, the principles could be generalised and applicable in a broader scope using other policy-based systems.

Typically, network resilience strategies comprise mechanisms for challenge detection and subsequent remediation. Most intrusion detection systems from commercial vendors such as Cisco [14], IBM [15] or Enterasys [16] are *signature-based*. Often these systems offer automated response based on detected signatures only, but in many cases require a human operator to interpret anomalous behaviour. *Anomaly-based* detection in real-time is resource intensive, and choices in the sampling strategy can impact its accuracy [17]. This may lead to high rates of false positive and false negative detections, thus restricting the automatic launch of remedies as they might create additional security risks. Our approach uses policies that can be carefully crafted and evaluated offline on a simulation environment. For a more extensive discussion on network resilience research, we refer the reader to [18], [1].

Policy-based management has been widely used for the configuration of network components and distributed systems [19]. However, configuring resilience mechanisms can present difficulties, especially when one considers the interaction between mechanisms for detection and remediation of challenges, and we propose the use of patterns to assist in the design of resilience strategies. Patterns incorporate ideas from software architectures, which typically define configurations of components and connectors as a means of structuring software development [20]. Patterns for the specification of policy-driven configurations were first defined in [4]. As part of our future work, we intend to investigate how resilience patterns can be pre-verified for conflicts, possibly using existing solutions and tools for policy analysis [21], [22].

VII. Conclusion

This paper has presented a framework and a process for the design and evaluation of network resilience management. The framework enables *(1)* the offline evaluation of resilience strategies to combat several types of challenges, *(2)* the generalisation of successful solutions into reusable patterns of mechanisms, and *(3)* the rapid deployment of appropriate patterns when challenges are observed at run-time.

We have developed a simulation platform to evaluate the performance of resilience strategies. The toolset is based on an integration between the OMNeT++ simulator and the Ponder2 framework. It supports the simulation of a range of challenge scenarios and the resilience strategies used to combat these challenges. The toolset enables us to identify best practices and the most effective policy configurations for challenges such as DDoS attacks, flash crowds and worm propagations.

Resilience strategies that perform well and can successfully address a particular type of challenge in the simulation environment can be ported to resilience patterns. These consist of roles for the same mechanisms used in the simulation environment, as well as the policy and event exchanges evaluated in the simulated strategies. Currently, the porting of a simulated strategy into a resilience pattern is done manually. Patterns can capture best practices and the experience of network operators into reusable configurations of resilience mechanisms, and can then be used to rapidly deploy a resilience strategy when challenges are detected in the live network. Moreover, specific mechanism instances and the threshold values to be used are defined at the point of deployment, according to up-to-date metrics. Therefore, resilience patterns are reusable and can cater for similar challenges that are manifest at different parts of the network, or for variations of a specific form of attack.

Acknowledgements

This research was supported by the European Union Research Framework Programme 7 via the ResumeNet project with contract number FP7–224619, the EPSRC funded India-UK Advanced Technology Centre in Next Generation Networking, and by NICTA (National ICT Australia).

References

[1] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks: Special Issue on Resilient and Survivable Networks (COMNET)*, vol. 54, no. 8, pp. 1245–1265, June 2010.

[2] Y. Yu, M. Fry, A. Schaeffer-Filho, P. Smith, and D. Hutchison, "An adaptive approach to network resilience: Evolving challenge detection and mitigation," in *DRCN'11: 8th International Workshop on Design of Reliable Communication Networks*, Krakow, Poland, October 2011, pp. 172 –179.

[3] E. Lupu, N. Dulay, M. Sloman, J. Sventek, S. Heeps, S. Strowes, K. Twidle, S.-L. Keoh, and A. Schaeffer-Filho, "AMUSE: autonomic management of ubiquitous systems for e-health," *Concurrency and Computation: Practice and Experience, John Wiley*, vol. 20(3), pp. 277–295, May 2008.

[4] A. Schaeffer-Filho, "Supporting management interaction and composition of self-managed cells," Ph.D. dissertation, Imperial College London, 2009.

[5] A. Schaeffer-Filho, P. Smith, and A. Mauthe, "Policy-driven network simulation: a resilience case study," in *SAC'11: 26th Symposium on Applied Computing*. Taichung, Taiwan: ACM, March 2011, pp. 492–497.

[6] M. R. Endsley, "Toward a Theory of Situation Awareness in Dynamic Systems," *Human Factors Journal*, vol. 37, no. 1, pp. 32–64, March 1995.

[7] P. Smith, M. Fry, S. Martin, L. Chiarello, M. Fischer, C. Rohner, G. Popa, H. D. Meer, A. Fischer, and N. Bohra, "New challenge detection approaches," August 2010, deliverable 2.2.b, ResumeNet. Available at: http://www.resumenet.eu/results/deliverables/d2.2b.

[8] G. Tadda, J. J. Salerno, D. Boulware, M. Hinman, and S. Gorton, "Realizing situation awareness within a cyber environment," in *Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, vol. 6242, no. 1, April 2006.

[9] M. Steinder and A. S. Sethi, "Probabilistic fault diagnosis in communication systems through incremental hypothesis updating," *Computer Networks*, vol. 45, no. 4, pp. 537 – 562, 2004.

[10] K. Twidle, E. Lupu, N. Dulay, and M. Sloman, "Ponder2 - a policy environment for autonomous pervasive systems," in *POLICY '08: IEEE Workshop on Policies for Distributed Systems and Networks*. Palisades, NY, USA: IEEE Computer Society, 2008, pp. 245–246.

[11] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *SIMUTools '08: Proceedings of the 1st International Conference on Simulation Tools and Techniques*. Marseille, France: ICST, 2008, pp. 1–10.

[12] D. Dittrich, "The tribe flood network distributed denial of service attack tool," University of Washington, Tech. Rep., October 1999.

[13] E. Asmare, A. Gopalan, M. Sloman, N. Dulay, and E. Lupu, "Adaptive self-management of teams of autonomous vehicles," in *MPAC '08: 6th International Workshop on Middleware for Pervasive and Ad-hoc Computing*. Leuven, Belgium: ACM, 2008, pp. 1–6.

[14] Cisco, "Cisco intrusion prevention system," Available at: http://www.cisco.com/go/ips. Accessed in: February 2011.

[15] IBM, "Security network intrusion prevention system," Available at: http://www-01.ibm.com/software/tivoli/products/security-network-intrusion-prevention/. Accessed in: February 2011.

[16] Enterasys Secure Networks, "Enterasys intrusion prevention system," Available at: http://enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx. Accessed in: February 2011.

[17] D. Brauckhoff, K. Salamatian, and M. May, "A signal processing view on packet sampling and anomaly detection," in *INFOCOM '10: Proceedings of the 29th Conference on Information Communications*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 713–721.

[18] P. Cholda, A. Mykkeltveit, B. Helvik, O. Wittner, and A. Jajszczyk, "A survey of resilience differentiation frameworks in communication networks," *Communications Surveys Tutorials, IEEE*, vol. 9, no. 4, pp. 32 –55, quarter 2007.

[19] M. Sloman and E. Lupu, "Security and management policy specification," *IEEE Network*, vol. 16, no. 2, pp. 10–19, Mar.-Apr. 2002.

[20] R. N. Taylor, N. Medvidovi, and I. E. Dashofy, *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, January 2009.

[21] M. Charalambides, P. Flegkas, G. Pavlou, J. Rubio-Loyola, A. Bandara, E. Lupu, A. Russo, M. Sloman, and N. Dulay, "Dynamic policy analysis and conflict resolution for diffserv quality of service management," in *NOMS '06: 10th IEEE/IFIP Network Operations and Management Symposium*, Vancouver, Canada, April 2006, pp. 294–304.

[22] R. Craven, J. Lobo, J. Ma, A. Russo, E. Lupu, and A. Bandara, "Expressive policy analysis with enhanced system dynamicity," in *Asia CCS '09: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*. Sydney, Australia: ACM, 2009, pp. 239–250.