DISco: a Distributed Information Store for Network Challenges and Their Outcome

Sylvain Martin*, Laurent Chiarello*, Guy Leduc*
*University of Liège, Research Unit in Networking, Belgium

***Abstract*—We present the design of DISco, a storage and communication middleware that enables distributed and task-centric autonomic control of networks. DISco allows multi-agent identification of anomalous situations (challenges) and assists coordinated remediation that will maintain service at an acceptable level, although degraded. The history of agents decisions, their context and outcomes is tracked as the situation evolves, and information is automatically gathered and organised to ease further human-assisted diagnosis.**

We then explore the feasibility of using state of the art peer-to-peer publish/subscribe and storage systems as building blocks for this service. The ability of those systems to support range queries and aggregation will be a key factor for their suitability to the task.

I. Introduction

Network monitoring mostly follow a location-centric, hierarchical processing [1] of information where most decisions are ultimately made by human operators. We argue that this model suffer three major limitations. First, the reaction of human operators is limited, especially when problems become as complex as nowadays botnet-driven denial of service or worm propagation. Second, the hierarchical approach allows some local manager to automate some "reflex" reaction based on local information. Any decision that requires knowledge – even summarised – regarding a larger area has to be deferred to a higher level in the hierarchy. As a result, the top of the monitoring hierarchy becomes a strategic target to intentional attacks: if taken offline or overloaded, defence of the whole network becomes severely compromised. Finally, although the causes of service degradation are numerous, the analysis of a challenging situation is performed by a single program (e.g. an IDS) that must take into account all their possible variations.

In line with the Resilinet/R^2D^2+DR strategy [2], we argue that the role Internet now plays in our society and the evolution of challenging events stems for a generalised ability for the network to self-defend by activating *remediation* mechanisms (traffic filtering, rerouting, ...) that will sustain service in a degraded, but acceptable state. The need for a more resilient Internet suggests that decisions always originate from a local system, possibly even defining areas of the network as Self-Managed Cells [3]. A *challenge*, in this context, is an event that impairs operation of the network and therefore threatens the quality and/or availability of the services it delivers. This definition includes malicious attacks, mis-configurations, accidental faults and operational overloads.

978-1-4673-0269-2/12/$31.00

Appropriate autonomic response to network challenges stems for *real-time* onset detection that acts as a background task, and triggers on-demand more sophisticated mechanisms involved in root-cause and impact analysis. This two-phases approach is crucial for saving resources of the network operator, as the (relatively) computationally expensive tasks can be focused both in time and in the amount of traffic they consider.

We argue that this multi-stage processing of monitoring information to ultimately provide a high-level understanding of a situation is similar to the process of hand-script recognition in architectural sketches [4] and could therefore benefit from a multi-agent approach, where each agent is specialised (task-centric) into one kind of challenge and can identify it with a certain confidence level. Translating this approach to network control, however, requires a distributed middleware that can efficiently relay information and self-document its decision to allow human-driven revision of the policies.

This paper collects the requirements for such a middleware (section II), proposes a service model (sections III) – the Distributed Information Store for Challenges and their Outcome – that fulfil these requirements and review the available components it could be built on (section IV). We further validate the concept on a DDoS detection scenario (section V) and briefly present the findings of our early feasibility simulations (section VI).

II. Problem Statement

We base our work on the findings of [5] regarding the detection of challenging situations in a network, that is, beyond *onset challenge detection*, which is usually achieved through anomaly or signature-based detection, additional steps are required to *classify* the challenge and understand its impact on the network and on ongoing communications. The ultimate goal of the challenge identification process is to activate and deploy a specific *remediation mechanism*, with all the configuration parameters defined, to handle the challenge and restore the service to a degraded, but acceptable level. If we consider defence against DDoS attacks as an example, onset detection could consist of link and queue monitoring. On-demand identification would involve a volume anomaly detection that pin-point the victim of the attack, and remediation mechanism would be a rate limiter for a firewall.

Figure 1 illustrates the communication chain involved in detection, identification and remediation of challenges through reports published into DISco. Probes present in forwarding plane components may vary in complexity, from single-

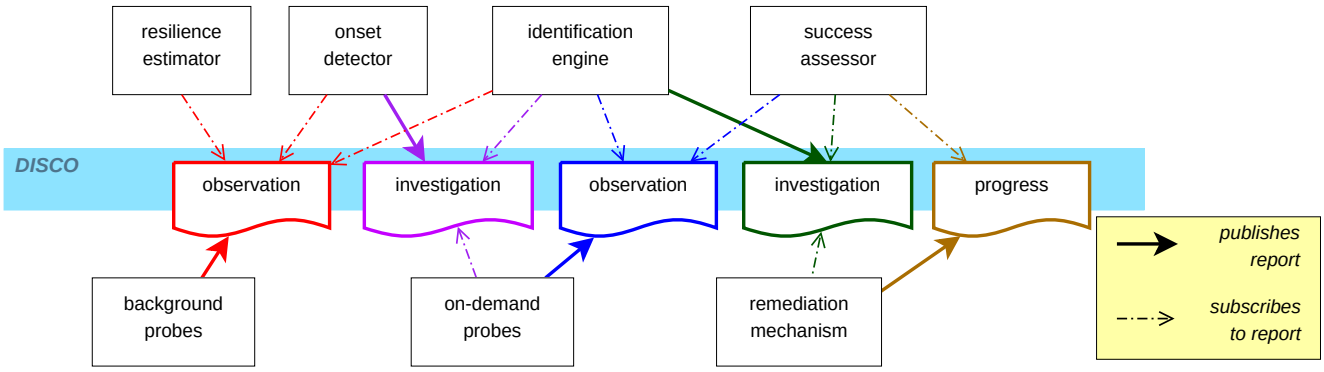

Fig. 1. Overview of the processing chain in detection and remediation of a challenge

variable monitoring (e.g. number of pending TCP connections) to more sophisticated entropy-based systems. Their reports are gathered by onset detection agents and identification engines, usually from multiple sources, which in turn produce investigation reports when the rise or decay of a challenge happens.

Notifications of challenges trigger the activation of mitigation components which will take actions to remediate the challenge. Because these decisions are taken automatically and require a response time that no human operator can offer, it is important to record the progress of the challenge as a whole (triggering conditions, evolution of the impact, side-effects) so that the fitness of the automated solution can be analysed later, and that response thresholds and parameters can be adjusted. In addition, each remediation mechanism has a companion *success assessor* agent that can validate or roll back the decision and acts as a real-time safe guard for the system.

We identified three key problems that hinder the development and deployment of efficient detection and remediation techniques, and we suggest that a common information dispatching and storage system is the proper abstraction to address them:

- There may be many probes, reporting more information than we can afford to relay on the network. Because we expect that multiple algorithms will be deployed, each operating as an autonomous agent to identify a specific kind of challenge, we want the probes to remain unaware of the number and identity of their listeners. Moreover, the relative network location of agents and probes impacts the accuracy we need on the information we receive.
- Detection, remediation and diagnostic actions are delayed from sensing activities. Yet, they may require detailed information on past events that preceded a trigger. Therefore, the required lifetime of individual events is hard to predict, but it requires careful management given the potential amount of generated information.
- New components will be deployed over time, to better identify and remediate unforeseen and foreseen challenges. They will likely alter the coupling between data by introducing new relationships and attributes. While this is an essential feature to guarantee successful evolution of machine-learning algorithms beyond their initial programming, it also implies that no database schema can be established in advance. Yet, the dynamics of information makes identifier-based solutions such as IF-MAP [6] not applicable "as is".

To address these problems, the Distributed Information Store for Challenges and their Outcome (DISco) provides the following features:

- an aggregation-capable publish/subscribe function that relays information between probes and agents, that can accommodate bandwidth restrictions;
- a distributed (peer-to-peer) storage system that provides system-wide, longer-term persistence for data that have been "elected for diagnosis";
- an annotation system, that extends information collected in the store, to further classify published information and adjust its lifetime accordingly.

The service we present in this paper assumes that devices are under the control of a single[1] (virtual) organisation, with appropriate secure channels established between nodes. Traffic on these channels is isolated from the regular traffic, but bandwidth is significantly reduced.

III. DISco Service Model

A. Data Model

Information is published in DISco as a *point* in a multi-dimensional space, where each dimension corresponds to an attribute of the published record, such as timestamp or destination address. A special "undef" value allows records to omit some attributes (e.g. an ICMP-related record has no "destination port" value), which is crucial for supporting the layered model of protocol stacks. This selection of *flat tuples* of *atomic values* over the more popular log lines will significantly simplify the creation of agents based on machine learning.

Subscriptions and lookup requests express their interest towards a set of records by means of a *region* defined by

[1]Guidelines on the interconnection of multiple domains are given in [7].

a set of ranges for a subset of the dimensions. By integrating the pub-sub and the database into a single service, we provide a unified access to agents to both past (lookup) and future (subscribe) events.

The absence of a primary key among the attributes is essential to the co-existence of agents that operate on the same data, but with different missions. Reports from a network-level honey pot[2], for instance, could be used to identify both a network scan involved in some worm propagation [8] or the occurrence of a SYN-flood DoS attack using spoofed IP addresses [9]. We note, however, that both agents would use different requests to access meaningful information. Worm detection agents would look up for `udp.dest_port`$= x$ while DoS agents would use `ip.src_addr` $\in N$.

With data balanced among nodes according to a single key, at least one of those requests will have to be flooded to the whole distributed system, and contributions of the nodes will have to be further merged by the requester. This is basically the approach recommended in map/reduce[10] to maximise the throughput of a large computation cluster where inter-node bandwidth is cheap. In a deployment where nodes are distributed in Points of Presence (PoP) of a network infrastructure, however, this approach would quickly lead to the overload of management-dedicated bandwidth.

B. Events Hierarchy

The *type* of an event uniquely identifies the set of attributes that are present in a report for such an event. To enable evolution of the system, we propose to organise these events in a taxonomy tree based on "IS-A" relationships, drawing inspiration from *Vocabulary Specification Trees* (VSTs) described in the ANA monitoring framework [11]. A parallel VST can be used to assign numerical identifiers to attributes in an event.

Sub-classes of events can be used to refine the known causes of an event and provide the additional attributes that complete information in that specific situation. It allows, for instance, to distinguish drops due to traffic shaping from those due to transmission errors, including average link load for the former and signal strength for the latter. Further sub-classing beyond a standardised taxonomy would allow the inclusion of protocol or vendor-specific information. A subscription defined using the generic event type will automatically capture all the events published with a sub-class event.

We propose to achieve this by mapping sub-trees in the VST to a binary prefix (e.g. `CA:FE/16`), which eases the matching of identifiers present in publications. The protocol used to distribute the relevant portions of the VST belongs to the management of DISco itself and is beyond the scope of this paper. We suggest that it could be performed incrementally at device deployment time, independently for each administrative organisation.

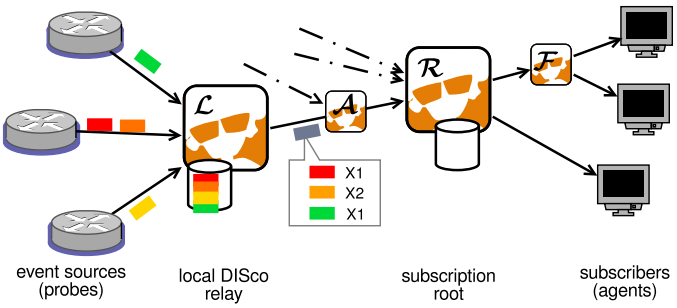

Fig. 2. Roles of DISco nodes in event aggregation & filtering

C. Filtering and Aggregation

By design, DISco does not require publishers to limit *a priori* the volume of reports they generate. This is motivated by the fact that publishers are mostly probes located in the forwarding plane and therefore should not embed assumptions on the use the management plane will make of published reports. To meet bandwidth limits, the pub-sub relies on adaptive filtering and aggregation of reports.

Filtering is essentially subscription-driven, and consists of discarding less relevant attributes or reports before their delivery to a subset of subscribers. It enables the presence of "pure payload" attributes that have no associated "dimension", and the merging of subscriptions. If a node receives two subscriptions S_A and S_B such that $S_A \subset S_B$, it can forward only S_B towards the responsible node and apply a filter that discards reports not matching S_A locally.

Aggregation, on the other hand, gathers several events and produce a single, coarser-grained notification. Both input events $e_1 \dots e_n$ and the aggregated event $e' = Aggr(e_1 \dots e_n)$ must be of an event type that is a sub-type of the one defined in the subscription[3]. Attributes of the events are aggregated independently, through the application of one of the pre-defined functions for their type. This implies that pub-sub nodes know the *type* of attributes for the event they relay, although they can ignore the specific semantic of the attributes. Subscribers control aggregation indirectly by expressing minimum and maximum desired event rates.

We assume that every event e produced by the system can be routed to a root node R_e that is the rendez-vous point for subscriptions that capture such events. Nodes upstream the root perform *aggregation* while nodes downstream the root essentially perform *filtering* and multicast the events towards subscribers. The typical processing of publications in DISco is depicted on Fig. 2.

D. Replies and Automatic Lifetime

The distributed store itself acts as a repository of previously published events, where lifetime of entries is automatically

[2]thus reporting that a datagram has been sent to an IP address that has no publicly advertised name.

[3]Aggregation thus differs from *correlation*, where the type of the generated event is usually different from the input event(s), and which happens in one of the agents interacting with DISco.

adjusted. It provides context information for on-demand agents and attempts to capture cross-layer view of challenges for "forensics" diagnosis by human experts. *Replies* to event provide a way for agents to highlight a subset of the events they received through their subscriptions and to promote them as part of the clues involved in the understanding of a higher-level process.

The first DISco node that processes a published event – the local relay L on Fig. 2 – is responsible for temporarily caching the event record before any aggregation is applied. The lifetime in this cache is typically only of a few overlay RTTs, to give subscribers the opportunity to reply to the aggregated event e' while keeping storage affordable.

When an agent considers that a specific event e' is important (e.g. because it reports a value over a detection threshold), it generates a reply message r that specifies an *annotation region* R_r and a set of *annotation attributes* A_r. This message is routed back to the individual caches L_i that contributed to the information present in e'. On each of these caches, we identify events $e_i \in (R_r \cap L_i)$ that intersect with the annotation region and transfer (e_i, A_r) to the root node R for storage.

A typical use of annotation attributes for an agent that e.g. starts an investigation process triggered by event e' is to establish a link between contributing events $e_1 \dots e_n$ and the newly generated investigation report v using its type and unique identifier as additional values.

We propose to implement replies support through a mechanism inspired from *zFilters* [12]. Each link between DISco nodes receives a unique identifier. When a report is routed to the root, the identifier of each link it crosses is added to the message's zFilter, which actually acts as a bloom filter. When two messages are aggregated at a node, their zFilters are ORed. To route a reply, the message is first handled to the subscription root. Then, at each node, the (reverse) neighbour table is then scanned, and a copy of the reply is sent on all the links that match the zFilter.

IV. Candidate Components

The resilience and self-organisation properties of *distributed hash tables* (DHT) provided the initial motivation for this work. However, the importance of range queries and multi-attribute searches in the service model quickly ruled out the possibility to directly build a prototype on DHT implementations. To illustrate the concerns in picking an appropriate substrate for the DISCo service, this section presents two DHT extensions and two alternative scalable distributed data structure that provide multi-attribute range searches and subscriptions.

Toronto Publish/Subscribe System (ToPSS) [13] is a Scribe [14] extension architectured around the concept of application-specific *schema* that defines how to map range queries over multiple attributes to hash-based topics. A set of search profiles (indices) is defined, that indicates which attributes of the entries must be matched. Attributes for which range queries are intended are sliced into pre-defined intervals.

The number of topics per subscription – and hence performance of ToPSS – is directly linked to the "quality" of the schema, which needs to be defined by an application expert. Given that DISco is intended to ease integration of agents and devices developed independently, without assumptions on the exchanged traffic, this dependence on external expertise is not advisable.

PastryStrings [15] suggests an almost opposite approach to range queries over DHTs. Searches are based on lexicographic strings and use virtual *tries* distributed over a Pastry DHT. A mixed routing approach is used, where some links between trie nodes are references to Pastry's own routing table. As a result, PastryStrings enables both prefix- and suffix-based searches at the cost of a higher response time for queries.

Multi-dimension queries are only supported through separate processing of the attributes: the source of an event has to retrieve and correlate subscription lists that match the event to be published. As a result, we argue that PastryStrings is better suited to systems where both publication rate per source and attributes count per event are low, two constraints that are unlikely to be met in the case of DISco.

Mercury [16] equally offers multi-range queries, and has already been demonstrated useful for autonomic systems [17]. However, it suffers an important drawback: a separate control structure (hub) must be maintained per attribute, potentially implying data replication. To route a query, Mercury uses the hub where the query is the most discriminating, and responsible node(s) on that hub performs additional filtering. With the amount of attributes envisioned in DISco, keeping the hubs synchronised in the presence of information updates would lead to an unsustainable overload.

SkipTree [18], in comparison, has native support for multi-range queries, and is organised as a distributed, self-optimising and high-dimension version of a quad-tree. Every node is responsible for a part of the data space that is defined by planar equations, organised along a shared decision tree. Although it is still possible that requests needs to be split during forwarding, the internal mechanisms allow automatic adjustments of the areas under control so that the average number of nodes involved per query remains low.

Unlike previously mentioned protocols, there is no public release of the SkipTree protocol known to date. Yet, we can note that, for a fixed set of partitioning equations[4], the SkipTree routing algorithm can be emulated with Scribe by setting the number of bits per digit to one.

V. Concept Validation : DDoS Detection

A. Network-Network Interaction Scenario

We illustrate the way DISco works on the DoS attempt depicted in Fig. 3. Attackers target link L that is required to reach a victim attached to V. They additionally identified that traffic towards destinations attached to U also uses this link and, thus, uses addresses U_i as well to dilute the signature of their attack. We also assume that attackers decided to have

[4]that is, assuming no optimisation or churn occur during experiments.

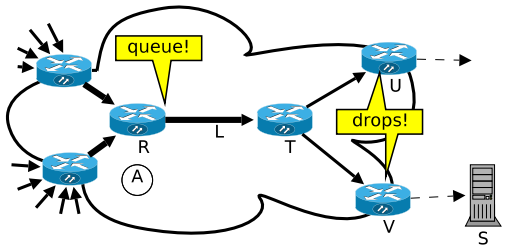

Fig. 3. Network under DDoS challenge, L being is the overloaded bottleneck link, $R-T-V$ the path to the victim through the network

their attack traffic dropped a few hops after L[5] in order to further evade detection by systems in V's network. This, however, makes their traffic look singular in the network containing L.

Routers in this example monitor the amount of IP packets that are rejected by the forwarding process, including those who have their time to live exhausted. They report this through `event.network.drops.forwarding.rfc791-ttl-exceeded` which contains (as attributes) flow identification (made of source/destination addresses, ports and transport protocol, in the case of IPv4), location of the reporting router and timestamp of occurrence. These events are published by V and U in the distributed store. Similarly, an important amount of "queue full" events occurring at R will trigger the execution of DoS-detection algorithms local to R such as identifying destinations of largest flows. A further analysis process A that was dormant in a system close to R previously subscribed to "any heavy-flow report event" from R, and possibly other routers in the same point of presence (PoP).

Upon reception of heavy flows reports from R, A will additionally subscribe to events reporting network-related errors downstream from R, enabling collection of reports from T, V and U. We assume here that A is an "expert" software agent that looks for and identifies the specific kind of DDoS attack we described above. The publish/subscribe mechanism fully allows multiple similar systems to execute concurrently and perform their own analysis using the same initial events. The conclusion of A that a DDoS is the cause of L's overload is published back in the store, with an attached confidence level, that enables appropriate remediation actions at R (or upstream) routers, and inhibits actions from the automated traffic engineering system.

B. Highlighted DISco Features

- *Local holding of data:* information about packets dropped at V and U are put under the control of DISco, but not yet transferred to a remote system until interest in such information is expressed through a `subscribe` call. Yet, it is important that such data can be looked up a posteriori, for instance when process A tries and gathers recent past statistics to figure out the dynamics of the challenge. Temporal aggregation can still be applied to reduce the available granularity of information over time.
- *Selective subscription:* while A needs extra information from T, U and V, it is only interested in information related to a fraction of the traffic those routers forward. For instance, if it identified 4.2.0.0/16 to be the destination of heavy hitters, it will add a constraint on attributes stating that `attribute.flow.rfc791-destination-address` must match that range.
- *Aggregating events from multiple sources:* A typically makes no difference between reports coming from U and V as long as they match the filter. This also highlights the need for describing a region of the network through an attribute constraint.
- *Flexibility:* A could broaden its monitoring criterion by subscribing to `event.network.drops*`, and receive notification of packet losses in the network regardless of whether they are due to TTL issues, congestion (queue full or early notifications), broken link, unknown destination, etc. This relieves A from knowing the actual network protocol stack details (as an ICMP snooping agent would have to) and allows monitoring of events generated by different layers as needed.

[5]by means of their Time To Live field or any similar hop count limit

	4 bits	3 bits	2 bits	1 bit
sent	28635	28630	28622	28640
subscribed	17509	17606	17556	17517
at root	1904	1597	1694	1933
ratio	9.2 : 1	11.0 : 1	10.7 : 1	9.1 : 1

TABLE I
AGGREGATION RATIO OVER SCRIBE, DEPENDING ON BITS/DIGIT

VI. Simulation Results

In order to evaluate the benefits of in-network aggregation of monitoring events, we modified the Scribe protocol implemented in the OverSim package [19], given the similitude of Scribe and SkipTree routing. We simulated reports of a honey net of 512 pots using the CAIDA "telescope" data set [20]. Aggregation happened solely between publishers and core nodes to ensure a maximum rate of 100 events per second per node[6]. The network is then completed by 32 listener nodes that will subscribe to /16 address blocks picked amongst the heavy hitters in the dataset. This simulates the situation where on-set detection already took place, and where honeypots reports are used to identify the nature of the challenge.

Table I reports, for each digit size[7], the number of reports sent during the simulation, the number of reports that have a matching subscription and the number of messages that have to be processed by subscription roots. We observed aggregation ratio of 9:1 to 11:1 with no strong effect of the digit size.

We further used this scenario to evaluate the potential of the *reply* mechanism. Link identifiers are derived from Scribe

[6]a hand-tuned rate to cause observable, but not systematic, aggregation on this data set.

[7]which controls the number of neighbours in Scribe

(160-bit) node IDs, and the first km bits of the LinkID are used to set k bits in a 2^m-bit zFilter. With $m = 10$, 99.8% *reply* messages visit no more than 5% of extra links compared to the corresponding publications. We varied k from 8 to 16 bits sets per link, with no visible effect on the performance for well-behaved publications. It does impact the performance for the outliers that may hold hundreds of reports aggregated in a single message. We suggest that this should be addressed with a more appropriate scheduler on aggregating nodes, rather than by fine-tuning bloom filter properties.

VII. Conclusion and Future Works

In this paper, we have presented the design of an integrated event dissemination and storage system that meets the need of challenge detection system both in terms of real-time and bandwidth-limited notification and context lookups with variable granularity. We conceptually illustrated its abstractions (pub/sub, vocabularies and aggregation) on the scenario of identifying and tackling DDoS attacks.

The aim of our design is to provide a unifying middleware through which task-specific agents (IDS, traffic engineering systems, server farms, etc.) can be informed of each other's decisions and adapt their automatic reaction accordingly, while offering the human operator a comprehensive report of activity as soon as the system fails to maintain the minimal service level. This differs from the typical resource availability monitoring problem for which DHTs have been applied successfully [21].

The use of dynamic aggregation as events are forwarded is a fundamental feature of our solution, and we confirmed its benefit on a well-known peer-to-peer pub/sub implementation. We also provided a preliminary feasibility simulation of the original *reply* mechanism that enables transition of events towards a long-term storage facility.

We have highlighted that protocols based on DHT have short comings that make them inadequate to the task, although they have been successfully used in specific cases, such as intrusion detection [22]. Protocols based on multi-dimensional space such as SkipTree are conceptually the ideal building block for the envisioned service. To compensate the lack of public release for the SkipTree protocol, an in-house prototype implementation has been started, and its performance on the scenario proposed in this paper is currently under study.

The adaptive storage that we coupled with notification forwarding is intended to provide the necessary information for challenge diagnosis and remediation refinement. Actual mechanisms that allow this store to adjust partitioning when new agents are deployed and that perform the necessary "garbage collection" to guarantee continued operation are still subject of future work. These future developments will open the door to the demonstration of our proposal on a complete detection and remediation scenario.

Acknowledgements

This work has been partially funded by EU project ResumeNet, FP7-224619. Sylvain Martin acknowledges the financial support of the Belgian National Fund of Scientific Research (FRS-FNRS).

References

[1] D. Durham, "RFC2748 the common open policy service protocol," Jan 2000.

[2] J. P. 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*, vol. 54, no. 8, pp. 1245 – 1265, 2010.

[3] A. Schaeffer-Filho, P. Smith, A. Mauthe, D. Hutchison, al. Y. Yu, and M. Fry, "A framework for the design and evaluation of network resilience management," in *13th IEEE/IFIP Network Operations and Management Symposium (NOMS 2012 – to appear)*, April 2012.

[4] R. Juchmes, P. Leclercq, and S. Azar, "A multi-agent system for the interpretation of architectural sketches," *Special Issue of Computers and Graphics Journal*, vol. 29, no. 5, 2006.

[5] M. Fry, M. Fischer, M. Karaliopoulos, P. Smith, and D. Hutchison, "Challenge identification for network resilience," in *Next Generation Internet (NGI), 2010 6th EURO-NF Conf.*, June 2010, pp. 1–8.

[6] "Trusted Computing Group (TNC) IF-MAP binding for SOAP specification version 1.1," May 2009.

[7] S. Martin, L. Chiarello, and G. Leduc, "Disco: a distributed information store for network challenges and their outcome," *CoRR*, vol. abs/1201.3073, 2012. [Online]. Available: http://arxiv.org/abs/1201.3073

[8] P. Porras, H. Saidi, and V. Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points," Computer Science Laboratory, SRI International, Technical Report, 2009. [Online]. Available: http://mtc.sri.com/Conficker

[9] D. Dittrich, "The "Tribe Flood Network" distributed denial of service attack tool," University of Washington, http://staff.washington.edu/dittrich/misc/tfn.analysis.txt, Technical Report, 1999.

[10] J. Dean, S. Ghemawat, and G. Inc, "Mapreduce: simplified data processing on large clusters," in *In OSDI 04: ProcS. of the 6th conf. on Opearting Systems Design and Implementation*. USENIX, 2004.

[11] V. Goebel, B. Gueye, T. Hossmann, and al., "Integrated Monitoring Support in ANA (v1)," 6th Framework Program - Situated and Autonomic Communications (SAC), Tech. Rep. FP6-IST-27489, D.3.7v1, Feb 2009.

[12] P. Jokela, A. Zahemszky, C. E. Rothenberg, and al., "LIPSIN: line speed publish/subscribe inter-networking," in *Procs. of ACM SIGCOMM'09*. New York, NY, USA: ACM, 2009, pp. 195–206.

[13] D. Tam, R. Azimi, and H.-A. Jacobsen, "Building content-based publish/subscribe systems with distributed hash tables," in *In International Workshop On Databases, Information Systems and Peer-to-Peer Computing*, 2003, pp. 138–152.

[14] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," *Networked Group Communication*, pp. 30–43, 2001.

[15] I. Aekaterinidis and P. Triantafillou, "Pastrystrings: A comprehensive content-based publish/subscribe dht network," in *Procs. of the 26th IEEE Int. Conference on Distributed Computing Systems*, ser. ICDCS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 23–.

[16] A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: supporting scalable multi-attribute range queries," in *Proc. ACM SIGCOMM' 04*, 2004, pp. 353–366.

[17] H. V. Hansen, V. Goebel, T. Plagemann, and M. Siekkinen, "Resource adaptive distributed information sharing," in *Procs. of the 16th IFIP conf. on Networked services and applications*, ser. EUNICE'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 246–255.

[18] S. Alaei, M. Ghodsi, and M. Toossi, "Skiptree: A new scalable distributed data structure on multidimensional data supporting range-queries," *Comput. Commun.*, vol. 33, no. 1, pp. 73–82, 2010.

[19] I. Baumgart, B. Heep, and S. Krause, "OverSim: A Flexible Overlay Network Simulation Framework," in *Procs. of 10th IEEE Global Internet Symposium (GI '07), Anchorage, AK, USA*, May 2007, pp. 79–84.

[20] E. Aben, S. C. Avila, and K. C. Claffy, "The caida ucsd network telescope two days in november 2008 dataset." [Online]. Available: http://www.caida.org/data/passive/telescope-2days-2008_dataset.xml

[21] R. van Renesse and A. Bozdog, "Willow: DHT, aggregation, and publish/subscribe in one protocol," in *IPTPS*, 2004, pp. 173–183.

[22] C. Zhou, S. Karunasekera, and C. Leckie, "Evaluation of a decentralized architecture for large scale collaborative intrusion detection," in *Integrated Network Management*. IFIP/IEEE, 2007, pp. 80–89.