Resilience and Survivability for future networking: framework, mechanisms, and experimental evaluation

Deliverable number	1.4
Deliverable name	Cross-layer optimization and multilevel resilience
WP number	1
Delivery date	31/08/2010
Date of Preparation	16/2/2011
Editor	Radovan Bruncak
Contributor(s)	Radovan Bruncak, Nafeesa Bohra, Andreas Fischer, Steven Simpson, Justin P. Rohrer, James P.G. Sterbenz, David Hutchison, Hermann de Meer
Internal reviewer	Marcus Schöller, Christian Rohner

Summary

In the ResumeNet project, the $D^2R^2 + DR$ resilience strategy is being evaluated. A part of this work is an exploration of the usefulness of cross-layer optimization to help assure multi-level resilience in the network. Cross-layering has been envisaged as a support mechanism for the resilience framework. Extracting useful information from the protocol stack and providing essential information for the analysis component should be seen as one of the fundamental tasks that the resilience framework should be capable of doing. An essential goal for resolving the resilience puzzle is what information should be provided and where the information should come from. Having identified information which might affect the decision on remediation, this could serve as the basis to reason about effective resilience actions subsequently. By exploring this direction it should lead us to the information sources to be adopted in the framework. We suggest cross-layering, monitoring, and context as potential candidates for helping us to solve the puzzle: cross-layering as a candidate for obtaining information across the protocol layers, distributed correlated network monitoring for getting a view from the different angles across the network, and context for getting essential information from wherever is appropriate to inform the decision-making. Context information, indeed, is likely to come from outside the network, from the environment in which the network operates. Together, these three aspects constitute an information framework that, we propose, should form the basis for an implementation of the $D^2R^2 + DR$ resilience strategy. From this study, we have learned that a cross-layer sharing database design could be the most applicable approach for building resilient networks. Next, the use of task-centric monitoring tools could be more applicable for network resilience than device-centric ones. Finally, by considering three case studies, we see the way ahead for further research on the information framework.

Contents

1 Introduction

Information analysis to help inform decisions on challenges and remediation steps involves several activities, starting with measurement and data collection, and moving on to processing that information. The ability to provide information that can affect specific decisions in the detection or remediation phases might be seen as crucial to the resilience framework. Challenge detection clearly relies on information analysis. Challenges to network operation can have an impact in various locations in the network, so it is important to have information facilities placed at strategic locations that allow collection and processing of data, both cross-layer and cross-domain.

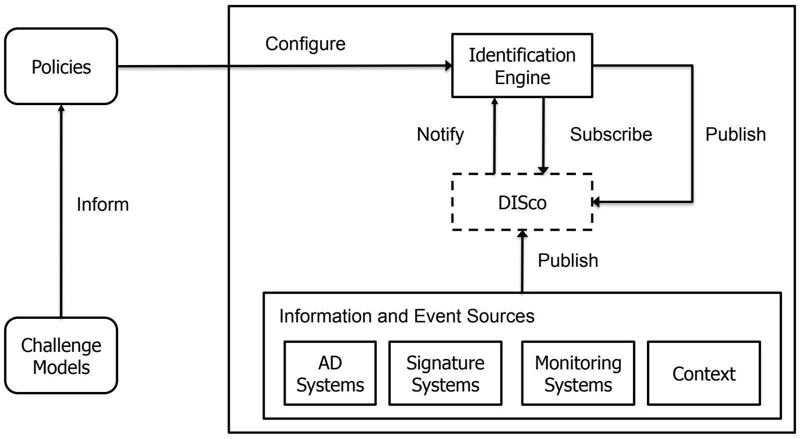


Figure 1: A challenge identification architecture [SFM+10]

Figure 1 shows the basic building blocks of Challenge Detection in the ResumeNet model. Cross-layer information, information obtained from monitoring and measurement, and context might be in use in the resilience framework. Information is collected throughout the network stored in the Distributed Information Store (DISco). The Identification Engine then subscribes to information provided by DISco and performs the necessary analysis over data. This analysis will be parameterised based on Challenge Models and Policies from outside the Challenge Identification Architecture. The results of the analysis are then fed back to the information storage, providing input for remediation actions. Together, these components form an information framework that we propose should be an implementation foundation to realise the $D^2R^2 + DR$ resilience strategy.

In the following sections, we aim to explore information sources for network resilience, formulate requirements for the information framework, and demonstrate applicability of the suggested requirements by several case studies – BGP misconfiguration, a cross-layer design example, and weather disruption in millimetre wireless mesh networks – before making final concluding remarks from the study.

2 Information Sources for Resilience

In this section, we explore various information sources for resilience. We argue that correct information is important to make effective decisions about how to mitigate a challenge. Without correct information, inappropriate remediation decisions might be made. Conflicting resilience objectives, or poor cross-layer design, might result in side effects causing the system to become more degraded, rather than improving resilience when applying remedies.

We provide a summary of information sources: cross-layering approaches, in which information is obtained from the protocol stack and provided to an adaptation component; context, which represents a broader view on cross-layering, in which any external information that might help to make better decisions is used; and monitoring, where information at multiple protocol levels is obtained via passive or active measurement being selected and deployed in the network.

In the following section, we present a cross-layer survey from the perspective of network resilience. We discuss several cross-layer architectures, their benefits and drawbacks for network resilience. Furthermore, we explore available cross-layer interactions to provide the required information from the protocol stack to decision components. Subsequently, we focus on cross-layer design methodologies, which might be used in the design phase of the resilience components requiring information from the protocol stack. Finally, the formal verification of cross-layer designs is discussed.

2.1 Cross-layer Survey

This survey reviews cross-layering research from the perspective of network resilience. Previous cross-layer surveys, such as [SM05, JTWW05] focus on interactions mostly for wireless networks. Cross-layering has been introduced as the breaking down of the strict layering boundaries that conceptually exist in the current network protocol stack. Information which is out of the scope of an engine in a single protocol layer might be provided to it by the cross-layer system. By using a cross-layer mechanism, a protocol's performance might be improved.

A cross-layer interaction might involve upward or downward information flows, whereby a protocol instance at a higher layer might be notified of conditions in the underlying network, or a protocol instance at a lower layer might receive suggestions from an upper layer on requirements for the adaptation. For example, a TCP sender could distinguish between errors on the wireless link and network congestion, based on signalling information from routers in the network [STP03]. The link layer might receive the delay requirements from an application to process delay sensitive packets with the required priority [GG01].

2.1.1 Cross-layer Interactions

Several cross-layer interaction designs can be observed from the literature:

- *upward information flow*, in which cross-layer information is propagated from a lower protocol layer to a higher protocol layer;
- *downward information flow*, in which cross-layer information is propagated from a higher protocol layer to a lower protocol layer;
- and an *interaction loop*, in which upward information flow depends on downward information flow, and downward information flow depends on upward information flow.

Inter-layer communication (upward/downward information flows) could be carried out by adding Hints and Notifications (HAN) [LUO02]. HAN can enable spectrum-efficient data handling, which can improve service quality for individual users and utilisation of wireless links. Hints might improve flexibility for application designers by allowing data with different requirements to be placed in the same packet. By sending notifications from lower layers, applications can react more specifically to the underlying problem. To simplify deployment and avoid interfering with existing Internet nodes, HAN can be implemented by using IP options and ICMP messages.

Collaboration between protocol layers (interaction loops) might be exploited by adaptation components in the protocol stack. For example, NADMS (Network-Assisted Diversity Multiple Access) [Kum85] has been proposed for collision resolution. An interaction loop between MAC and physical layer has been suggested to recover packets from collisions based on estimation how many users have collided. Consequently, certain retransmission might be requested.

2.1.2 Cross-layer Architectures

Several cross-layer architectures have been proposed to allow information sharing. Two categories can be observed from the literature: the shared database, and direct communication between layers.

Shared Database Using a shared-database approach, protocol instances access cross-layer data via an external database that exists in parallel to the protocol stack on each node in the network. The shared database introduces new interfaces between protocol layers. An optimisation program can run on top of the shared database, to modify parameters of the protocol instances as a response to the triggered adaptation.

Another shared-database concept has been suggested by Chen et al. [KSK02]. The cross-layer plane is a distributed data-accessibility service that allows access to multimedia data within a group of cooperating nodes that share information with each other. Data advertising, lookup, and replication services have been suggested to achieve data accessibility, which are the main components of that cross-layer framework.

Defrawy et al. [Def06] propose a similar approach to the cross-layer plane. The architecture is based on a cross-layer server that communicates with a protocol at a given layer by means of a client. Interaction occurs between layers by a client at a layer sending an event to the cross-layer server. The event is then forwarded to the target layer(s) by the server, where the targeted client will receive the event from the cross-layer server and conduct certain adaptations. The framework allows the cross-layer server to invoke an active optimisation control.

DISco (Distributed Information Store) [SFM^{+}10] has been suggested as a shared database for resilient networks. It provides basic functionalities for data subscription and data publishing which might be useful to a resilience component to get required information from the network about the state of the infrastructure.

Direct Communication Between Layers An alternative to using a shared database involves allowing direct communication between protocol layers. The inter-layer signalling pipe approach [WR03] can be used for propagating signalling messages layer-by-layer, along with packet data flow inside the protocol stack. Signalling information can be associated with a particular packet entering or leaving the protocol stack. This approach of layer-by-layer message propagation could prove inefficient because of unnecessary processing overhead, and message formats tend to be inflexible. The CLASS framework [WR03] aims to address these limitations. It allows two protocol layers to communicate directly through out-of-band cross-layer signalling shortcuts implemented using ICMP messages. However, the management of shared memory space is an issue that needs to be resolved.

From the perspective of the resilience principles proposed by Sterbenz et al. [SDE+10], which are presented in Figure 2, arguably the shared database approach is the most suitable for resilience, as it reduces interaction complexity (one of the resilience principles related to tradeoffs) and makes the operation of a networked system more tractable. However, it does potentially lead to a single point of failure and a target for attackers, unless special measures are taken.

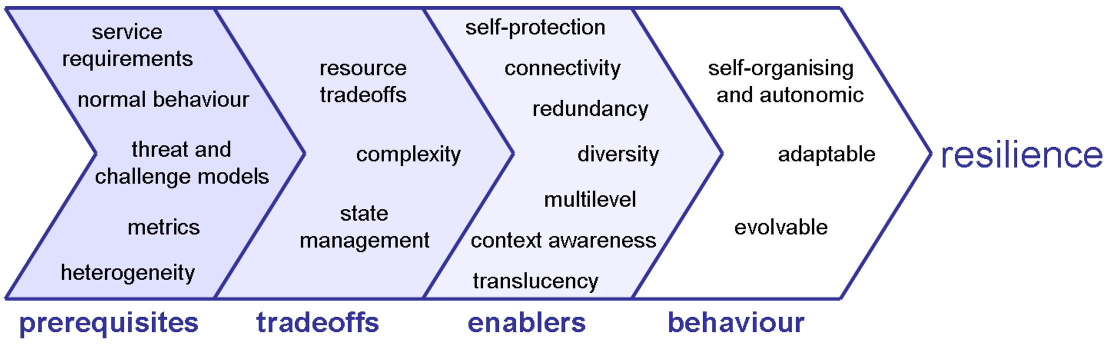


Figure 2: The ResiliNets principles

2.1.3 Cross-layer Design Methodologies

A methodological framework has been proposed for network design and verification by Papachristodoulou et al. [PL04], in which TCP/AQM (Active Queue Management) schemes are compared. The authors emphasise that it is important to construct robust models for the modules, as this will capture the uncertainty in modelling and component parameters that might be considered in the design processes. The authors have developed a tool for analysing such systems based on non-linear control theory for optimisation schemas.

Baldo et al. [NM07] have suggested using fuzzy logic for the representation of cross-layer information, and the implementation of control strategies where the search for optimal settings might be conditioned by parameters such as affordable complexity, scalability, or modularity. The authors used fuzzy logic as a knowledge-representation scheme for cross-layer information, and fuzzy-logic control theory as a technique for implementation of cross-layer optimisation strategies.

A cross-layer optimisation foundation has been introduced by Fu et al. [FM07] which aims to allow each layer to make autonomous decisions individually, while maximising the utility of the wireless user by optimally determining what information needs to be exchanged among layers. Because the wireless user interacts with the environment at various layers of the protocol stack, the cross-layer optimisation problem can be formulated as a layered Markov decision process, in which each layer adapts its own protocol parameters and exchanges information with other layers to co-operatively maximise wireless performance.

Cross-layer optimisation can be a complex problem, for instance, joint optimisation in network congestion control [JJM10, JP04]. The task is not trivial to resolve and a number of approximation solutions have been proposed. To address the complex cross-layer optimisation problems, the cross-layer design might be decomposed and resolved into separate pieces. A mathematical theory has been proposed for doing this [MSCJ06]. However, the theory is based on Network Utility Maximisation (NUM) [FMD98], which targets network congestion control cases only.

Further work is necessary to understand the complexities and tradeoffs to be made with various cross-layer designs and use of mechanisms at different layers for resilience. Initial work presented in Section 3 on a cross-layer formalism could be used as a starting point to understand these tradeoffs.

2.1.4 Formal Verification of Cross-layer Designs

Several cross-layer designs have been proposed. It is not clear which approach is the most applicable. Would it be the shared database or direct signalling approach? Cross-layer mechanisms could be compared according to cost-benefit analysis, design complexity and stability. It has been shown that a bad cross-layer design might cause a degradation of the system [KK05], so a cross-layer design should be formally verified to assess its quality according to a set of criteria.

Initial work on formal verification has been carried out by Kliazovich et al. [DMF08], who have compared cross-layer signalling mechanisms. The work is limited to performance metrics only, and does not consider fundamental verification of cross-layer design. For example, in order to formally verify stability of a cross-layer design, the system as whole should be evaluated, including the likelihood of the system to become unstable, for instance. This can be applicable for the case of rate-adaptive MAC protocols [KK05] in order to configure robust cross-layer design. Thus, the formal verification of cross-layer designs could be done in a way in which the components of a design are formally verified separately and then the system as whole.

From the literature [SM05] on cross-layer design, three categories of cross-layer components can be identified: *generators* of cross-layer information; *protocols* for distribution of cross-layer information; and the *processing components* of the delivered cross-layer information. Formal verification of generators could focus on the evaluation of the impact of cross-layer information on the other protocol layers. Formal verification of distribution protocols could focus on evaluation of the feedback control. It has been shown that information that is lost at the feedback to an adaptation component can cause that the system to become unstable [WMS01]. Formal verification of the processing of received cross-layer information could focus on a mechanism to resolve conflicting objectives.

Separately, each component could be evaluated with concern to local stability. As resilience agents in a computer network might adapt the characteristics of the controlled nodes to specific changes, the dynamicity of the adaptation is worthy of study. For example, frequently changing adaptations triggered by suggestions from cross-layer information might affect the local service. If a node adjusts the modulation quickly then the system is more vulnerable to destabilisation, as in the case of rate-adaptive MAC [KK05]. The system as whole might not react to the changes appropriately, despite the verification of local components as successful. Therefore, the global stability of the system (a wider view on the system) should be considered in the formal verification.

2.1.5 Related Cross-layering Activities

Cross-layering has been applied in a number of application domains, and it could be argued that it is no longer just considered as a mechanism for optimising the performance of a protocol design. Cross-layering techniques have been used for anomaly detection, and cognitive and autonomic networking, which we discuss here.

Cross-layering has been discussed in wireless cognitive networking [YC08] in order to efficiently allocate resources and improve quality of service. A blackboard model has been presented for coordinating multiple cognitive nodes at run-time, determining which nodes receives information from other cognitive nodes, and updates actions with conclusions and suggestions. A structured message format is suggested for an abstraction for parameter values, which are optimised with the respect to an objective function.

Cross-layering has been discussed in autonomic networking [MSP07]. Cross-layer information can be utilised to attain self-behaviours [XXHL05]. An effort has been made to utilise cross-layer information for self-optimisation and self-healing respectively. An information sensing and sharing framework [MMD07] has been suggested as a support for cross-layering in autonomic networking with upward abstraction of the information flow as its the cross-layer design objective.

It has been suggested that anomaly detection systems could exploit cross-layer information in order to achieve lower false positives in wireless ad-hoc networks where there are resource constraints and lack of a centralised control [GSR06]. The authors present a decentralised monitoring IDS for detecting jamming attacks at lower protocol layers by evaluating the varying channel and network dynamics. The cross-layer design incorporated in the IDS helps it differentiate the malicious jamming behaviour from a network failure.

2.1.6 Summary

We have discussed the various approaches to cross-layering, identifying their suitability for building resilient networked systems and outlining areas where further research is necessary.

We propose that a shared database architecture tends to be more applicable for resilience compared to cross-layer signalling designs. Manageability and the ability to provide information are main factors in the comparison. However, shared database architectures do not provide any active component, such as an active control function for tuning internal parameters of a protocol layer, yet active components might be seen as fundamentals for resilient networking.

In some cases, direct signalling between protocol layers might be required, such as in highly dynamic wireless networks [RJPS08], and one signalling approach could be a based on piggybacking. However, the manageability of cross-layer interactions or management of shared memory space might be a problem. A particular design might be unsystematic and "spaghetti-like", and cause stability problems. Thus, considering these arguments together with the resilience principles [SDE+10], the cross-layer shared database might be a more applicable approach, compared to cross-layer signalling, for network resilience.

Furthermore, during the design phase of a cross-layer mechanism, a number of difficulties might be experienced, such as the mentioned stability problems. Cross-layer design methodologies might deliver robust cross-layer designs in order to avoid undesired side effects, but the current methodologies focus on network congestion cases only, while network resilience considers a wider view of challenges. Further work is necessary to understand the complexities and tradeoffs to be made with various cross-layer designs and use of mechanisms at different layers for resilience. Initial work presented in Section 3 on a cross-layer formalism could be used as a starting point to understand these tradeoffs.

2.2 Monitoring

This section aims to review network monitoring, and focuses on the importance of distributed correlated network monitoring from the perspective of network resilience. We explore relevant concepts and differences between available monitoring techniques, followed by a discussion on the importance of distributed correlated network monitoring. A comparison between available monitoring tools is given and an overall summary concludes this section.

2.2.1 Network Monitoring

Network monitoring is an essential component and helps to control the devices connected in a network by collecting and analysing data from them. Network monitoring is becoming increasingly difficult due to the increasing use of heterogeneous computing environments. It is becoming even harder with users' increasing expectations for resilience, reliability and QoS. Conventional network monitoring is based upon Simple Network Management Protocol (SNMP) [LC90]. SNMP is often used in centralised network environment and gives flexibility to the network administrators to manage the network from a single point. Centralised network monitoring has some drawbacks, such as a lack of scalability, excessive processing load at the manager, excessive bandwidth usage, etc. An alternative is distributed network monitoring where the centralised monitoring strategy is replaced by interoperable monitoring systems and hence, provide a global picture of the network for diagnostics.

Conventional Network Monitoring Network monitoring mechanisms are used to collect and analyse network traffic data for network management. At present a number of network monitoring/diagnostic tools are available, but many of them are limited to specific protocols. Different network monitoring tools have been developed and used by different entities depending upon their specific needs. For example, there are monitoring tools that collect time series system data from a set of nodes and allow operators to identify hotspots, and high-level bottlenecks such as disks, CPUs, or network failures. Similarly, there are monitoring tools which are used to collect "health data" from hosts, services, and network components. In summary, monitoring should give an insight into a running system, and provide system developers and administrators with a facility to spot and resolve failures. This is an important aspect for any network monitoring tool in today's complex and distributed network environment. Especially for large distributed systems where there is a possibility that multiple components may fail independently. In such cases, it is important to record and examine faulty executions and also to understand the cause of unexpected behaviour.

Our focus in this section is particularly upon the techniques that directly collect executional data from the systems. Commonly, these tools are based on techniques such as logging, tracing and profiling. Based upon the above mentioned techniques a monitoring tool can be classified as problem detection and problem analysis/diagnosis. Problem detecting tools are used to identify the problems, whereas problem analysis/diagnosis tools try to establish the root cause of an identified problem.

Logging is the simplest technique used for reporting the state of a running system. Logging provides visibility into a running system by making use of print statements either to a console or to a file and often requires processing. The log can be treated both as a history and as a source for determining the misconfiguration of the system. Logging is mainly considered as a technique to detect problems but can also be used for accounting, forensics, information retrieval.

Compared to logging, which is mainly used by system administrators, tracing is a technique for problem analysis as it tracks down the misbehaviour of the running systems. To our understanding a trace is a sequence of events that is well structured and comprehensive, which means that it can capture for a certain time span all relevant events and allows reconstruction of a sequence of events at the desired level of granularity. Hence, a trace can be a subset of a log. For example, a trace can be a sequence of web server events, which can be extracted from the server logs. The logs may contain additional information that is not part of the trace, such as error information. In other words, it can be said that tracing is a more structured form of logging since it records sequences of execution of basic building blocks of a program. Hence, tracing is task-oriented while logging is application oriented. Tracing is extremely valuable for performance debugging, trace-driven simulations, performance tuning, auditing, validating models, etc. Tracing can be performed at different levels of granularities, such as network packets, file system activity, machine instructions. There is a very thin line between tracing and logging. Depending upon the application, tracing may generate a large amount of data that may not all be needed, or may result in an overhead, in that case a better solution is to make use of profiling.

In contrast to tracing (which produce the sequences of events) profiling is a set of related techniques that aggregate information about resource usage. Profiling is helpful for analysing the performance of certain components of a system. It can be said that profiling is useful in performance tuning. For example, with profiling one can determine which components of a system are consuming most time, or which activity of a program is consuming most of the resources. Profiling may be useful in production mode where it can be used to determine the overall performance of the system (measuring response time or web pages/data base queries). This information is particularly helpful if gathered at regular intervals to compare the results over certain time intervals or to measure response times during peak and low-load hours.

Distributed Network Monitoring In a distributed system, it is quite difficult to understand how ansy given client request is being fulfilled within a service and why other requests fails. For this reason, we need distributed network monitoring. Distributed network monitoring involves multiple "pollers" distributed around the network, reporting on events from multiple locations in the network. With single point network monitoring one can see the network from a single perspective. With distributed network monitoring one can see the network from a number of different views across the network, i.e, it provides a global view of the network. Distributed network monitoring therefore allows for a deeper understanding of the network, giving us the ability to detect outages and bottlenecks more easily.

Internet applications are becoming increasingly distributed and complex and so the challenge of developing, deploying, managing and troubleshooting them is also increasing. In order to deal with all the aforementioned problems and to make the current network more robust and resilient against challenges, distributed correlated network monitoring mechanisms are required.

Distributed correlated monitoring involves the use of multiple points (cross-layer, cross-domain) distributed across the network for measuring network performance. Distributed correlated network monitoring is important because it helps in the collection of useful information across different network layers and administrative domains of the network so that the network can be managed and controlled using the collected information. Such network monitoring techniques are required to allow network monitoring applications to check the state of their network devices. In order to achieve the tasks mentioned above, it is important to make use of monitoring tools which are task-centric rather than device-centric. Unlike device-centric monitoring tools, task-centric tools enable an operator to causally trace the complete execution of a networked system across the boundaries of applications, protocols, and administrative domains. Furthermore, as more and more network devices are used to build bigger/complex networks, network monitoring techniques need to be expanded for monitoring the network as a whole. Distributed correlated network monitoring is required because:

- Network devices are becoming increasingly complex and distributed
- Network infrastructure is becoming more complex because of interrelated services

In light of the above statements, we argue that causal, end-to-end (cross domain) task oriented monitoring must be an integral part of network services to enable resilience. On the one hand with such a monitoring mechanism it is possible to:

- Follow a service request from start to end, with variable level of details across applications and network layers
- Log the relevant information from the devices connected with each tagged operation, which can then be reported back and provide a comprehensive view of what network operations have been executed as part of the task.

On the other hand task-centric monitoring tools:

- Can provide causal, end-to-end (cross domain) visibility into network services and hence enable the discovery of a number of bugs and help in diagnosing the performance faults
- Can be incrementally deployed and should be compatible with components provided by different parties, much like services themselves

2.2.2 Comparison between Available Distributed Network Monitoring Mechanisms

At present a variety of network monitoring tools focus on monitoring network status, aggregating data across devices and layers, but many of them are limited to a specific layer or application. For example, monitoring tools used to trace the route between two end system does not provide cross-layer information, hence provide limited visibility. Other monitoring tools provide monitoring at individual machines or at network interfaces, but failed to provide an end-to-end (cross domain) picture of a specific task. In this section, our aim is to make a comparison between available distributed network monitoring tools.

Consider Table 3, which illustrates the comparison between available distributed correlated network monitoring tools. The tools are differentiated with regard to whether they are task or node based, they are cross-layer or cross-application, and if they offer correlated/causal behaviour and are either open source or are commercially available. Based upon the comparison of Table 3 in the following section we will describe the pros and cons of each tool and then justify why distributed correlated network monitoring is important from the perspective of resilience.

Monitoring tools like HP Openview[1], SNMP (Simple Network Management Protocol) [LC90] and Netflow[2] are node based monitoring tools. HP Openview network node manager is a proprietary network management tool. It makes use of SNMP data to retrieve the

[1]`http://www.osalt.com/openview-network-node-manager/`

[2]`http://www.cisco.com/go/netflow/`

Table 1: Comparison between Network Monitoring Tools

Tool	Node/Task oriented	Cross-Application	Correlated / Causal	Cross-Layer	Availability
NetFlow	Node	No	No	Mainly focuses on IP flow information	Open Source
HP Openview	Node	Yes	No	Yes	Commercially available
Traceroute	Task	No	No	No	Open Source
IP Traceback	Task	No	No	No	Several Schemes
ARM (Application Response Measurement)	Task	Yes	No	Limited to application layer	Open Source
Pip	Task	Yes	Does not capture cross-layer correlations	No	Conceptual implementation
Causeway	Task	Yes	No	Yes	Conceptual implementation
SDI (Stateful Distributed Interposition)	Task	Yes	No	Yes	Conceptual implementation
NetReplay	Task	Yes	No	Focused on application and transport layers	Conceptual implementation
Splunk	Task	Yes	Yes	Yes	Commercially available
Pinpoint	Task	Yes	Yes	Yes	Conceptual implementation
Magpie	Task	Yes	Yes	Yes	Conceptual implementation
X-Trace	Task	Yes	Yes	Yes	Open Source

information it requires. HP Openview has an ability to coordinate views at different granularities and is also able to coordinate network policy changes. SNMP is a protocol developed to manage nodes (servers, workstations, routers, switches, hubs, etc.). SNMP is used for collecting information from, and configuring network devices such as servers, routers, switches, etc. on an IP network. It lets operators inspect instrumentation data from network devices such as packet count and error conditions. Similarly, Cisco IOS Netflow runs on Cisco IOS enabled equipment for collecting IP traffic information, and is mainly used for visualisation, accounting and traffic analysis. It performs monitoring at the routers and switch interfaces. Monitoring tools like SNMP, Netflow, HP Openview are cross-application and distributed but do not provide correlated/causal behaviour.

Traceroute [Mal93] as a diagnostic tool is often used for network troubleshooting. It shows the route over the network between two systems, listing all the intermediate routes a connection must pass through to get to its destination. Traceroute is useful where IP connectivity problems are concerned, but it does not deal with proxy servers and DNS failures. Furthermore, it may also help to identify routing problems or firewalls that may be blocking ICMP traffic. However, Traceroute information has been frequently used by hackers as a way to acquire sensitive information about a company's network architecture. For this reason, many networks block Traceroute requests, or de-prioritize the ICMP time exceeded message that is required to determine round trip time. Traceroute gives detailed information about the path taken between two end systems but it does not cross network layer boundaries, thus provides limited visibility. It is task based and open source but it does not fulfil the rest of the criteria given in Table 3.

Another approach used for network monitoring called IP Traceback [Alj03], is a method which reliably determines the origin of the packet on the Internet. IP is a trusted protocol where the source IP address is not authenticated and because of this the source address in an IP packet can be falsified (IP address spoofing) which may be used in DoS (Denial of Service) attacks. IP Traceback is a mechanism that not only identifies the network path that has been traversed by attack traffic without requiring interactive support from Internet Service Providers (ISPs) but also provides 'forensic' information after an attack has completed. Several approaches namely: ICMP Traceback messages, Probabilistic Packet Marking [SWKA00], Hash-based IP Traceback [SPS+01], etc. are in use. All the above mentioned schemes are task based and distributed but have their focus on the IP layer hence do not support cross-application and correlated/causal behaviour.

Application Response Measurement (ARM) [Joh98] is an API jointly developed by an industry partnership (Tivoli System (IBM) and Hewlett Packard). It is used to monitor the availability and performance of applications. Using ARM, an application can be managed for availability, service level agreements, and capacity planning. ARM performs monitoring from the perspective of the application itself, so it reflects those units of work that are important from the perspective of the business. Applications define units of work (transactions) that are meaningful within the application. Typical examples are transactions initiated by a user, or transactions with servers. Applications then call the API when transactions begin and end, allowing these transactions to be measured and monitored. ARM is targeted at the application layer and its focus is to diagnose performance problems in nested transactions.ARM is an open standard

Pip [RKW+06] is a path-based debugging approach for distributed systems. Bugs may create discrepancies between a system's actual behaviour and the behaviour expected by the programmer, and the Pip approach compares actual and expected behaviours of distributed systems and exposes structural errors and performance problems. It allows programmers to express their expectations of the system's communication structure, trimming, and resource consumption by means of a declarative language. It makes use of system annotation and instrumentation tools to log actual system behaviour, as well as query and visualization tools for exploring expected and unexpected behaviour. Pip is specifically useful for systems driven by user requests, since it captures the timing (delay) and resource consumption associated with each request. It is used to find unexpected behavior in each application, and helps to isolate the causes of poor performance and incorrect behavior. Pip targets single distributed applications under the same administrative domain and does not capture cross-layer correlations.

Causeway [CEC05] and SDI (Stateful Distributed Interposition) [RS04] provide operating system constructs that the application can use to track its activity in a multi-tiered system. A multi-tier application is comprised of multiple program components communicating among themselves to execute incoming requests. In such applications, a request is executed by multiple threads of control on different application components. The threads of control exchange data among themselves along communication channels. Many applications, for example, web sites generating dynamic content and web service applications, have a multi-tiered structure. Causeway performs automatic propagation of metadata across system-visible channels (channels that are implemented in the OS kernel and system libraries) such as pipes and sockets. The main difference between Causeway and SDI is that Causeway provides an API to be called from application code to perform metadata propagation across system-opaque channels such as shared memory, whereas SDI doesn't have this support.

In order to diagnose the root cause of a problem, a multitude of monitoring tools/methods are in use, but a more recent approach, called NET-Replay [AA09], has been developed by Annand et al. It helps application end points conduct in-band investigation of the glitches they encounter. Before forwarding packets, network elements need to remember the list of packets that they have successfully forwarded in a past time-window. This is done by packet-marking at the interface. When flow senders detect that some glitch had occurred based on feedback from the receiver via duplicate acknowledgement, timeouts, reception errors in RTSP etc., the sender re-transmits, or replays, a small number of additional packets which are exact duplicates of the packets that had experienced the glitch. When the re-played packets arrive at the receiver, the annotations in them can be used to derive the type and location of the glitches encountered by the original packets. The whole process takes a number of extra RTTs, which is a disadvantage. At present, the Net-Replay primitive can be used by end-hosts and applications to characterize network-induced anomalies, including packet drop, reordering and excessive queuing delay. Net-Replay is a task-based approach, but it does not provide correlated/causal behaviour. It is an application- and transport-protocol-driven framework for diagnosing network glitches, and is a conceptual framework.

Splunk[3] is a commercial system that aggregates logs from different sources in a network and make use of information retrieval techniques to index and provide interactive searches on multiple logs. It crawls logs, metrics, and other data from applications, servers and network devices and indexes it in a searchable repository from which it can generate graphs, SQL reports and alerts. The main advantage of Splunk is to assist system administrators in the identification of patterns and the diagnosis of problems. Log files can be correlated across systems and software components which can help administrators to find out the cause of system failures. The main disadvantage of Splunk is that it is unlikely to work across organizations, and is not guaranteed to have the relevant causal connections.

[3] `http://www.splunk.com/`

Pinpoint [CKF+02] is a problem-determination framework and used to detect faults in large distributed systems. Pinpoint provides a coarse-grained tagging of numerous real client requests as they travel through the system. It make use of data mining techniques to correlate the success and failure of the request and to determine which component(s) are most likely to be at fault. It is a modified J2EE middleware used to capture the paths that component-based Java system took through that middleware. By analysing the components which were involved in a failed request but are not part of the successful request the authors are aiming to provide high accuracy with relatively low number of false positives. Pinpoint monitors at middleware level, so it requires no knowledge of the application component. However, it therefore cannot distinguish the deterministic failures due to pathological inputs from other failures, for example, a user experiencing bad cookies causing consistent failures.

There are two main limitation of this approach:

- It is not able to differentiate between the set of components that are tightly coupled and are always used together.
- Pinpoint does not work with faults that corrupt state and affect subsequent requests. The non-independence of requests create difficulty in diagnosing the real fault because the subsequent requests may fail while using a different set of components. For example; a user trying to login but failed because the component responsible for creating a new account have stored a wrong password.

Magpie [BDIM04] is a tool chain that works with events generated by middleware, OS, and application instrumentation. Magpie has an ability to capture the control path and the resource consumption demanded by the application request as they are serviced across components and machines in a distributed system. The main advantage of this approach is that within Magpie both the functional progress of the request and the resources consumed by that request are recorded at every stage. Hence, it gives a complete picture of how requests have been served, for example, what modules were touched and where the time was spent. Was the request served from a cache or did it cause a disk access? Per request data can be analysed and is helpful to construct a concise model of the workload and anomaly detection. However, Magpie correlates events in different components but focuses mostly on single system or distributed systems that are highly instrumented in a compatible way which is the main limitation of this approach. Pinpoint [CKF+02] and Magpie [BDIM04] have conceptual implementation.

X-Trace [FPK+07] can be described as a framework that reports the message and protocol header flow between network layers, nodes and applications that support the framework. Supporting applications and devices generate so-called X-Trace metadata which is transported forward in-band with the actual data communication of the application. When X-Trace [Fis10, Pet09] enabled implementations receive metadata-enriched packets, they generate reports on their state and actions taken which are sent out-of-band to either a centralized reporting server or a requesting party. X-Trace not only traces devices and layers but it also traces the actual path taken by the data message rather than trying to provide only snapshots of the network infrastructure as a whole. It provides a logging-like API for the programmer, allowing the recording of events during the execution. The distinguishing feature of X-Trace is that it records the causal relations among these events in a deterministic fashion, across threads, software layers, different machines, and potentially different network layers and administrative domains. X-Trace groups events into tasks, which are sets of causally related events with a definite start. Moreover, X-Trace enables independent local analysis of trace data for possible causes of failure. This gives network operators some control over the amount of tracing data they want to share with other parties.

2.2.3 Monitoring summary

Distributed correlated network monitoring is important for network resilience in order to identify faults and threats that are otherwise hard to track. In particular, challenges involving a combination of different services may pose a problem for troubleshooting. In order to diagnose the root cause of a given challenge, a multitude of monitoring tools/methods are in use, as discussed above, each having its own advantages and disadvantages. Cooperative correlation of events on different network layers is an important building block in uncovering complex challenges. However, the question how to properly correlate events remains an open topic. Moreover, the efficiency and overhead between different distributed network monitoring mechanisms still has to be evaluated.

2.3 Context Information

For the most part, network management decisions are made based on information that is obtained from monitoring systems in the network, such as those discussed in Section 2.2. However, to be able to make autonomic decisions about the nature of a challenge and how to respond to them through remediation, a broader range of information needs to be used. In addition to traditional network monitoring information, and that collected using approaches to cross-layering, we propose the use of *context information*.

A precise and meaningful definition of context is difficult to ascertain, with several different definitions of context having been developed. Many of these definitions are based on synonyms for context [HNBR97, FF98, RPM98]. Other definitions are based solely on examples of context [SAW94, BBC97]. As discussed by Biskop [Bis08], a useful definition of context was given by Abowd et al. [ADB$^+$99], which is based neither on synonyms nor on examples:

"Context is any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."

For our purposes, the *entity* that context information is being used to characterise is the networked system and services that must adapt to improve the resilience provided to applications and their users.

Context information has been proposed to be used in a number of application domains. The use of context in computer networks has been proposed for use in ubiquitous computing. CARISMA [LWC03] is a mobile computing middleware, which exploits the principle of reflection to enhance the construction of adaptive and context-aware mobile applications. It makes use of policies to describe how context changes should be handled. The authors argue that policies might result in conflicts, and a classification of conflict types has been stated. Furthermore, the authors argue that conflicts have to be resolved at execution time rather than resolved statically.

Goodall et al. [JWPA06] propose a tool that aims to provide a broader view on the data collected from the network. The idea is to analyse the packets within a broader context of surrounding network activities. Traditional tools are limited towards discovering and understanding patterns and anomalies. The proposed tool could give network analysts a simultaneous view of the bigger picture and individual packet details. The authors claim that the tool can reduce cognitive burden of analysts.

The ACAN (Ad hoc Context Aware Network) system architecture [MA02] has been suggested. In the architecture, a context manager agent interprets the information captured by sensors and processes it into higher-level context data to be used by an ad-hoc application. This process is used to minimise the user attention that is required and maximise the relevance of the information. ACAN introduces mechanisms for network configuration, QoS provisioning and dynamic adaptable applications. Efstratiou et al. [CKN+00] have introduced architectural requirements for adaptive mobile applications. The authors focused on the resolution of conflicts caused by the need to adapt to multiple contextual triggers in a cooperative environment.

Gold et al. [GM01] have suggested an approach based on context awareness to allow peer-to-peer applications to exploit information on the underlying network context to achieve better performance and better group organisation for mobile ad-hoc networks. The aims are to reduce overhead by exploiting information on resource availability, and to reduce battery usage by using context-aware strategies. Wang et al. [MLPY07] applied context-awareness to mobile peer-to-peer networks for ubiquitous learning. The authors claim that the learner's attention might be reduced by exploiting context information such as location, activity, and ambient information. Biskop [Bis08] proposed three levels of abstraction, namely context data, context information, and context knowledge content distribution networks where the context might be seen as "know-how", which could be obtained in two ways: either by transmission from another who has it (by instructions), or by extracting it from experience.

With regard to resilience, earlier work has demonstrated how the use of weather information, an example of context, improves the resilience of millimetre wave wireless mesh networks, which perform poorly in heavy rain [AJA+09]. In short, information regarding the trajectory of weather fronts is used to make pre-emptive routing decisions to avoid potentially affected regions of the mesh. Another use of context could include using information regarding news events to rationalise an unusually high-volume of network traffic that is causing problems. Anecdotal evidence suggests this form of information is used frequently in Network Operation Centres (NOCs) by human operators.

A key area for further work is to investigate the systematic use of context information for resilience. In Section 4, we discuss initial thoughts on an information framework that could be used to enable such systematic use of context information.

3 Cross-layer Framework

Cross-layering is not an end in itself, but it is an essential support capability in order for intelligent and adaptive mechanisms to operate effectively. It uses dials to provide information to higher layers, and knobs to influence the operation of lower layers. Making proper use of these will allow the transport layer to adapt to the operational condition of the underlying network, as well as modifying its mode of operation based on the requirements of the applications above it. These cross-layer knobs and dials will enable the selection and tuning of specific resilience mechanisms that are appropriate for the current network state.

For a given scenario, it can be apparent what cross-layer knobs and dials are needed. With this understanding it becomes an iterative process to reduce the complexity of the cross-layer control loops, which will involve re-evaluating the layer at which it is most appropriate to implement a mechanism. For example, consider a path diversification mechanism: the optimal use of paths requires information from both the routing and transport layers, and a path diversification algorithm itself may operate in either of the layers. By evaluating the cross-layer controls needed, we can determine which layer the functionality should reside in to minimise the complexity of the knobs and dials needed.

3.1 Cross-layer Formalism

When evaluating decisions regarding locating functionality at a given layer we are primarily evaluating the tradeoffs involved. Design complexity is one of the primary concerns in cross-layering, so we will evaluate the tradeoff of increased complexity compared to the increase in resilience. We have begun to formalise the representation of knobs and dials as follows.

The set of all knobs

$$\mathbb{K} = \mathbf{K} \cup \mathbf{k} \tag{1}$$

is the union of out-of-band **K** and in-band **k**. The set of all dials

$$\mathbb{D} = \mathbf{D} \cup \mathbf{d} \tag{2}$$

is the union of out-of-band **D** and in-band **d**. Knobs and dials are defined on the boundary between layers L_i and L_j where i and j are either numbers, e.g., {1, 1.5, 2, 2.5, 3, 4, 7, 8} or layer designators, e.g., {HBH, NET, APP}. An individual knob or dial between layers L_i and L_j is then $K_{i \to j}(\mathrm{desc})$ where 'desc' is a descriptor, e.g. Bit Error Rate (BER). Therefore the set of all out-of-band knobs and dials between layers i and j

$$\mathbf{K} = \cup_{\forall K \in \mathbf{K}} K_{i \to j} \tag{3}$$

represents a vertical relationship when $i \neq j$ and represents a horizontal relationship when $i = j$.

To carry this further, we can fully represent a layer n protocol instance at time t in terms of its knobs and dials, as well as its *state* $s(t)$ and *context* $c_n(t)$. For L_n, we can define state with respect to time as:

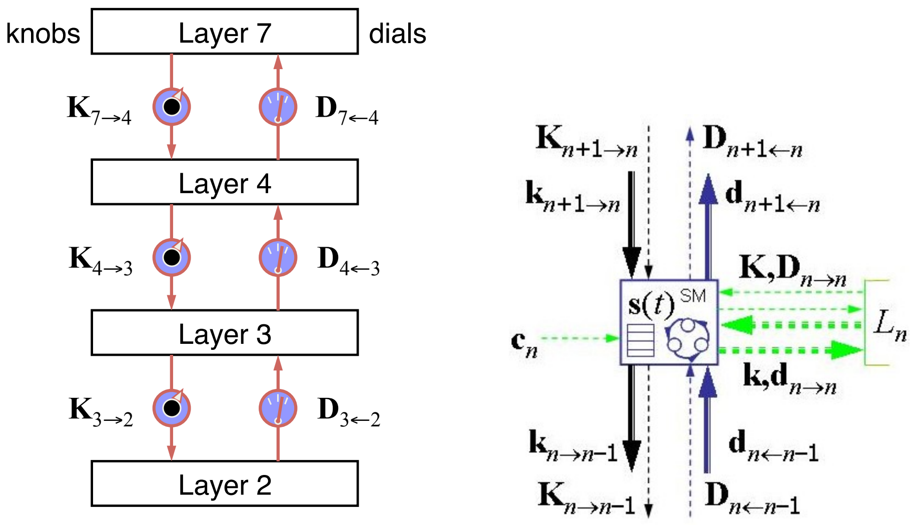
$$\mathrm{s}(t+1) = f(\mathbb{K}_{n+1 \to n}, \mathbb{D}_{n \leftarrow n-1}, \mathrm{s}(t), \mathrm{c}_n) \tag{4}$$

where f is a function specific to the internal algorithm of that particular protocol.

Figures 3(a) and 3(b) illustrate these relationships, which we are applying to mechanisms such as explicit cross-layer support for error control, see Section 3.2, which allow the network layer to notify the end-to-end layer of the cause of data loss. Explicit Loss Notification (ELN) notifies the transport layer of loss due to corruption, Explicit Congestion Notification (ECN) notifies it that loss was due to congestion, Explicit Outage Notification (EON) not only notifies it of loss but that there is an outage in the path, and Explicit Delay Notification (EDN) notifies the transport layer that data has been delayed but not actually lost. In addition to ELN, cross-layer notifications indicate that some corruption was experienced but it was recoverable using Forward Error Correction (FEC).

3.2 Case Study: Error Control

Error control falls into two classes: closed-loop (e.g. automatic repeat request, or ARQ), and open-loop (e.g. forward error correction, FEC).



(a) Multilayer overview (b) Detail

Figure 3: Cross-layer formalism

ARQ essentially involves checking that a packet was received correctly, and asking for it again if not, or (conversely) assuming that a packet was not received correctly, and retransmitting it unless told otherwise. A checksum or cyclic redundancy check (CRC) is usually employed to detect errors in the components of a packet. These can only detect that errors have occurred, and cannot identify exactly what change took place. ARQ incurs a cost in time for retransmissions, including waiting for acknowledgements, but increases the likelihood of successful transmission with more retransmissions.

FEC involves ‘armouring’ a packet transmission with additional data, or encoding the original data such that it takes up more space. If errors are within a certain threshold, the receiver can rebuild the original packet from the additional or encoded data. Depending on the particular FEC scheme, the threshold can be increased by supplying more additional data, or encoding in such a way that more space is required. FEC therefore incurs a cost in bandwidth in order to pay for greater assurance that the packet is delivered correctly.

FEC and ARQ are orthogonal techniques. FEC may help to reduce the effective packet error rate of a transmission path, while ARQ allows complete packet losses that exceed the level of FEC protection to be recovered with further effort. In our simulations, ARQ and FEC are independently controllable, and can both be switched off for baseline experiments.

Error control may be implemented on an end-to-end (E2E) basis, or on a hop-by-hop (HBH) basis. In E2E error control, ARQ acknowledgements are not generated until a packet is received by the ultimate destination.

The thesis of cross-layering is that applications (upper-layer entities) and routers/switches (lower-layer entities) co-ordinate their efforts to deliver payload by exchanging more information than is currently passed between layers. Applications will provide more information on the requirements for delivery (e.g. timeliness), while network fabric will report greater information about local and aggregated performance (e.g. error rates). Applications will use the lower-layer information to make better choices about E2E resilience, while the network will use upper-layer information to make better choices about HBH resilience.

By using information about low-level network performance, an application might choose to enable stronger E2E error control, or choose between ARQ and FEC according to the type of performance problem the network is suffering from. Similarly, the network can use hints from applications using its links to choose the type and strength of HBH error control. A first challenge exists in determining what information naturally available at one layer is useful to another.

Without cross-layering, the application does not normally have a network-layer view, as the network is presented as a simplified abstraction. Similarly, network nodes do not normally have an application-layer view – packet payloads are seen as opaque data. This presents a further challenge: getting information from one layer to a place where it is visible to another.

3.2.1 Error Control Mechanisms

In simulation, errors have three potential effects on a packet.

1. Errors occurring in a packet header will result in it being dropped. Attempting to use information in the header would be unwise, as there is no longer any confidence in (for example) the destination address for subsequent forwarding, or in sequence numbers used for acknowledgements. If FEC is used, its type and strength might be corrupted, so even then the packet must be discarded.
2. Errors occurring in the payload that cannot be corrected by FEC might result in the packet being dropped. Such corrupted data is not wanted, and need not traverse the network further, nor be passed to the application.
3. Those same errors might be tolerated by the application. Instead of dropping the packet, it is passed on including the errors. Over several hops, many errors might accumulate, but might still be tolerated upon delivery to the application.

FEC is modelled in the simulation as a form of Reed-Solomon encoding. In this scheme, payload is split into chunks of a certain size, then each chunk is given a number of 'parity' bytes that make it up to a codeword of 255 bytes. These parity bytes are derived from the original bytes in the chunk, and allow the receiver to correct upto a certain number of byte errors in the codeword. For example, a codeword of 91 payload bytes plus 164 parity bytes can tolerate upto 82 (half of 164) byte errors distributed throughout the codeword. By varying the ratio of parity bytes to payload bytes in a codeword, the armouring against transmission errors can be controlled, at the cost of some overhead that results in a lower raw throughput.

At the HBH level, the number of byte errors in the payload of each codeword is calculated, as it is in the parity bytes. If these are added and remain within the threshold of the FEC strength, the chunk is considered to be recovered without errors. Otherwise, either the whole packet is considered lost, or that chunk is considered to have the errors simulated in the payload.

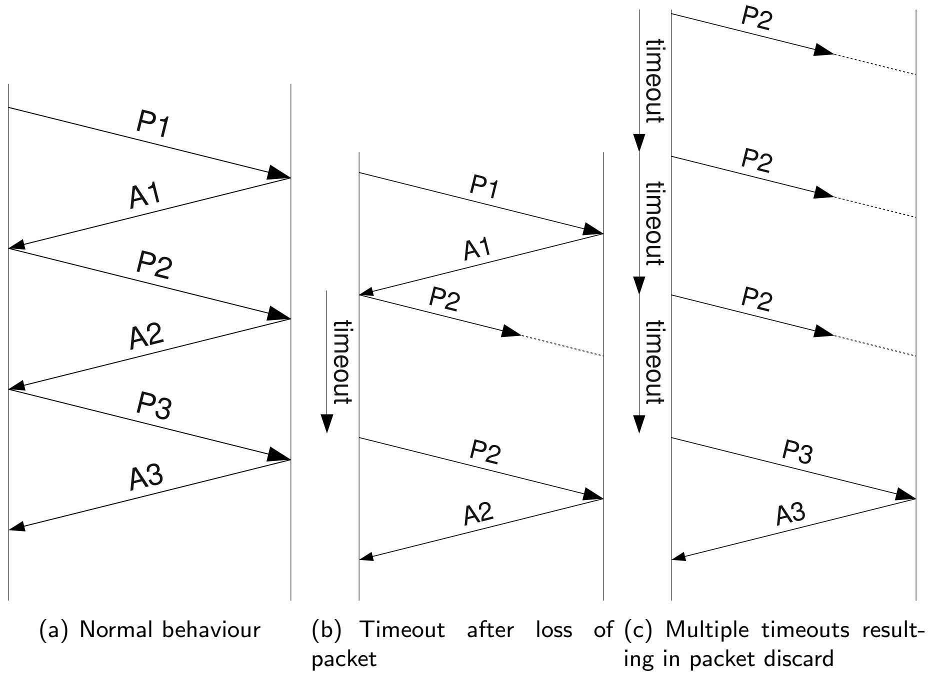
Such errors are tagged onto the simulated packet, and are considered by the E2E FEC mechanism. If there are too many for E2E FEC, the payload is considered lost.

The HBH FEC strength is a number from 0 to 127. Zero implies no FEC. 127 implies one byte of payload per codeword/chunk, plus 254 bytes of parity, tolerating 127 errors in the codeword. FEC strength can also be expressed as a percentage. 30 parity bytes plus 225 payload bytes tolerates 15 byte errors. Evenly spread across the codeword, that places $\frac{225\times15}{255}$ in the payload, yielding $\frac{15}{255}$, or about 5.9%. 127 corresponds to $\frac{127}{255}$ or just under 50%.

There are three acknowledgement modes: ‘none (i.e. disabled), ‘cumulative’, and ‘selective’.

ACK mode ‘none’ Each packet submitted for delivery is transmitted once, in order of submission. There is no attempt to retransmit the packet, and no acknowledgements are sent. This mode primarily exists to test FEC modes without any form of acknowledgement, but also serves as a baseline or control when used without FEC.

‘Cumulative’ ACK mode Each packet includes a sequence number (SN) (HBH: 32 bits), and is retained in a transmission window of limited size until acknowledged. An acknowledgement identifies a SN n, implying that everything upto n has been received. Reception of an acknowledgement therefore removes all window entries upto and including n. This mode is used for Go-Back-N (GBN) (figure 5) and Stop-and-Wait (S&W; figure 4) forms of ARQ; a window size of 1 produces S&W behaviour.



(a) Normal behaviour (b) Timeout after loss of packet (c) Multiple timeouts resulting in packet discard

Figure 4: Stop-and-Wait ARQ: (a) The receiver acknowledges each correctly received packet. The sender waits for acknowledgement before sending the next packet.

(b) When packet 2 is not delivered correctly, no acknowledgement is sent, so none is received, so the sender gives up waiting for the acknowledgement, and retransmits.

(c) When packet 2 is not delivered correctly several times, a threshold is reached, so the sender gives up, and continues with packet 3.

Each window entry includes a timeout and try count. When the timeout occurs, the try count is checked, and the entry is discarded if the count has reached a configured threshold (figure 4(c)). Otherwise, the packet is retransmitted, and all subsequent entries’ timeouts and counts are reset (figures 4(b) and 5(b)). Retransmission only occurs with the earliest window entry, as it always has the earliest timeout.

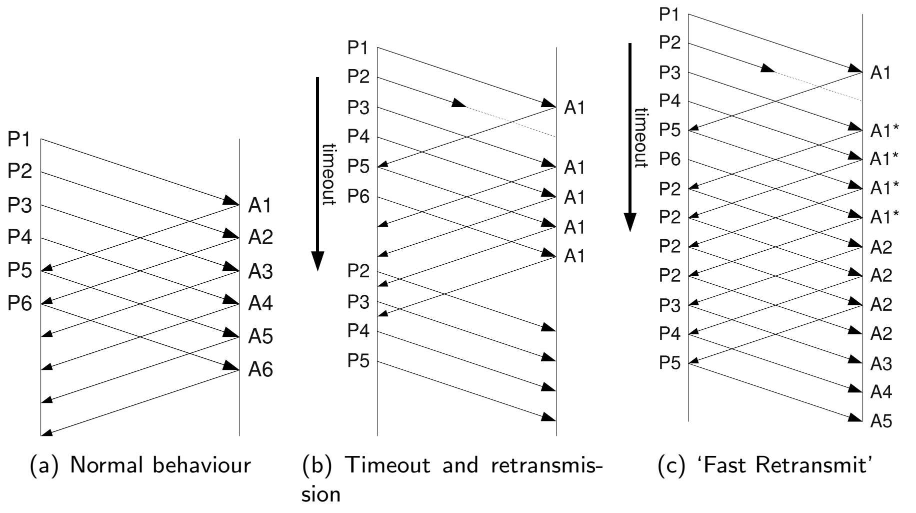


Figure 5: Go-Back-N ARQ: (a) With a window size of 4, packet 5 can begin transmission only when packet 1 is acknowledged successfully.

(b) Packet 2 is not delivered correctly, so no acknowledgement is sent. As a result, packet 3 is received unexpectedly, so packet 1 is re-acknowledged. Transmission stops when the window is full (from 2 to 6), and begins with packet 2 again when it has timed out.

(c) Using marked acknowledgements as a signal, the sender can start retransmitting the lost packet early.

The receiver keeps an expected SN e. If a received packet is not of SN e, the receiver sends an acknowledgement of $e - 1$ (figure 5(b)). Otherwise, it acknowledges e, increments e, and delivers the payload to the upper layer.

Acknowledgements can be aggregated. Instead of acknowledging e immediately, several consecutive SNs may be acknowledged with a single packet when a threshold is reached, or when an unexpected SN is encountered.

When a SN $n > e$ is acknowledged with $e - 1$, a 'repeat' flag is present in the acknowledgement. The sender can use this to determine when to perform a fast retransmit (FR), behaving as if the timeout for the window entry e has occurred (figure 5(c)). (Performing FR when the flag is absent causes GBN to degenerate into several parallel and duplicate S&W interactions.) The sender also keeps a count and a threshold of received 'repeat' flags. A threshold of zero implies that no FR is to be performed; otherwise, an FR is performed when the count reaches the threshold. The count is reset when a non-'repeat' acknowledgement is received.

When the packet for the first entry is (re)transmitted, a 'first' flag is present in the packet header. This flag tells the receiver that no SN earlier than that of the current packet should be expected, so it should increase its expected SN e to match the packet's SN. Without this, the receiver has no robust way to know that the sender has abandoned a packet due to too many retries.

'Selective' ACK mode Each packet includes a sequence number (SN) (HBH: 32 bits), and is retained in a transmission window of limited size until acknowledged. An acknowledgement identifies a SN n and a set of nearby subsequent SNs, implying that n and those

Scheme	ARQ mode	ARQ attempts	ARQ window	FEC strength
No-ACK	None	N/A	N/A	0%
S&W R255	Cumulative	255	1	0%
S&W R3	Cumulative	3	1	0%
GBN R255	Cumulative, Fast Retransmit	255	255	0%
GBN R3	Cumulative, Fast Retransmit	3	255	0%
SACK R255	Selective	255	255	0%
SACK R3	Selective	3	255	0%
FEC 2%	None	N/A	N/A	2%
FEC 4%	None	N/A	N/A	4%

Table 2: Simulated HBH Schemes

other SNs have been received. Reception of an acknowledgement therefore removes all window entries matching n or the other numbers in the set. This mode is used for SACK forms of ARQ.

Each window entry includes a timeout and try count. When the timeout occurs, the try count is checked, and the entry is discarded if the count has reached a threshold. Otherwise, the packet is retransmitted. Retransmission may occur with any SN in any order.

When the receiver receives a packet of SN n, it acknowledges n, and delivers the payload. This might result in multiple deliveries of the same SN, as well as out-of-order deliveries.

Acknowledgements can be aggregated. Instead of acknowledging e immediately, several SNs may be acknowledged with a single packet when a threshold is reached.

3.2.2 Impact of Choice of Error Control

Hob-by-hop simulations are performed over a single link, with an `OnOffApplication` sending 1024-byte packets to a `SinkApplication`. The rate of the link is 5Mbps, while the application sends at 1Mbps. The link delay is 2ms. The bit-error rate (BER) is varied between 10^{-6} and $10^{-2.25}$ to observe its effects on packet delivery ratio and one-way delay.

Nine schemes are tested, as shown in table 2. 'No-ACK' provides a baseline. Stop-and-Wait (S&W), Go-Back-N (GBN) and SACK each have a scheme in which undelivered packets are dropped after few (3) or many (255) attempts. Reed-Solomon FEC is tested with two strengths.

Figure 6 shows packet delivery ratios of the nine schemes. Initially, all do well with a BER of 10^{-6}, but 'No-ACK' quickly decays. All three ARQ schemes with few retries start to fall later but more quickly. The FEC schemes are then affected, but are overtaken by the three 'R255' schemes which drop sharply.

Figure 7 shows one-way delay, measured from start of a packet's original transmission to its delivery. Undelivered packets are excluded. The graph shows that 'No-ACK' and the FEC schemes incur no extra delay[4], while GBN performs better than SACK and S&W at the lower error rates. This is likely due to GBN's Fast Retransmit, allowing it to retransmit earlier than the other schemes, which depend on timeouts set longer than the round-trip time. At higher error rates, ARQ schemes have delays in the order of seconds, though the PDR is so low in the

[4]Note that encoding and decoding delay of FEC are not modelled in these simulations.

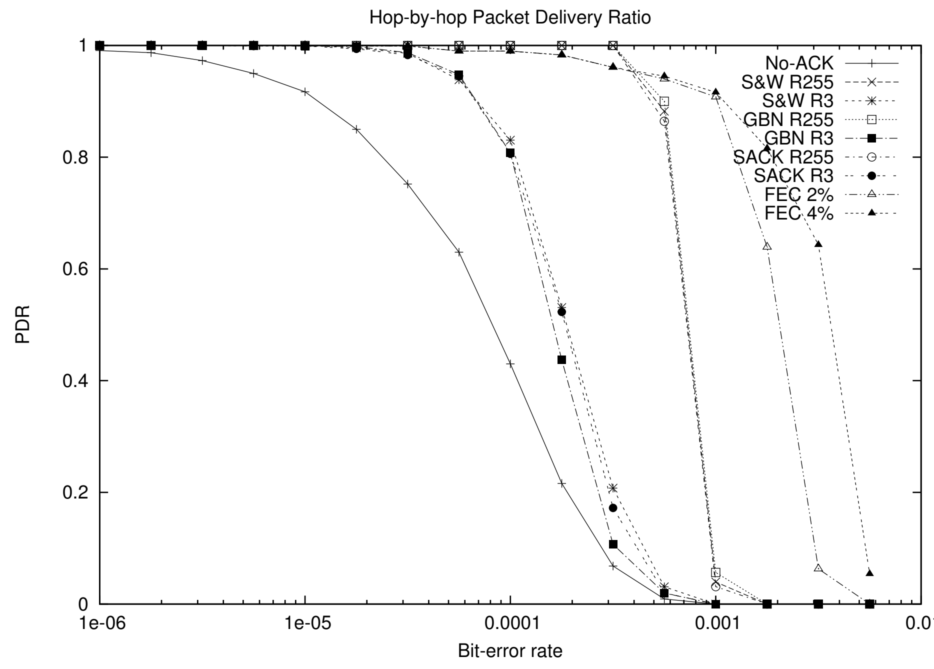


Figure 6: Hop-by-hop Packet Delivery Ratio (PDR)

extremes that few or no packets arrive, so measurements are either of poor statistical value or non-existent.

At the HBH layer, a network element could decide to enable more retransmissions if it knows that the applications using it need stronger guarantees of delivery and can tolerate longer delays. These requirements (strength of delivery guarantee and delay tolerance) are 'knobs' in the cross-layer framework.

Conversely, information about how packets were lost can be useful to the application. Knowing that most packet loss was due to congestion allows the application to understand that the network capacity is exhausted, and it should either back off, or seek alternative routes. If it were more due to corruption, the application could strengthen its E2E FEC and inform lower layers that it is more tolerant to errors, or it could request that lower layers increase their resilience to errors. The contribution of corruption and congestion to the losses of an application's streams are 'dials' in the cross-layer framework.

3.3 Conclusion and Outlook

The choice of open- or closed-loop error control at hop-by-hop and end-to-end layers serves as an example decision which can benefit from cross-layering. It presents an opportunity to study and understand the nature of cross-layer information, how to make use of it, and how to deliver it where needed, so we will use it as a basis for developing our cross-layering framework.

Future work should extend the simulations to explore the interactions between error-control schemes used simultaneously at different levels, to determine the conditions under which applications and network nodes should re-tune or change those schemes to optimise performance according to different criteria such as timeliness, throughput or reliable delivery. This will identify some of the knobs and dials that need to appear in our framework, specifically those supporting optimised error control, and help us to understand the nature of knobs and dials in

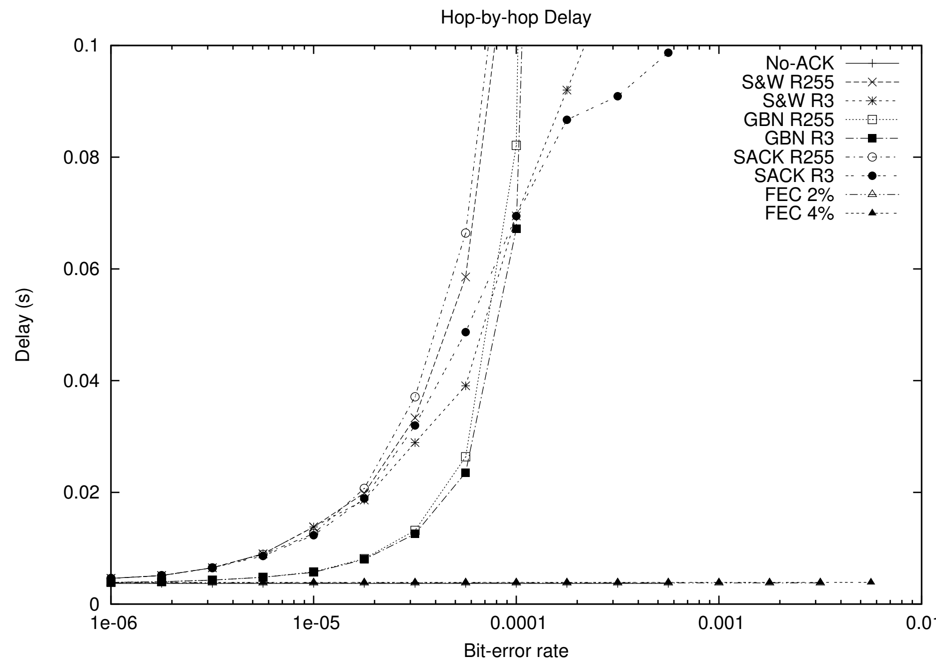


Figure 7: Hop-by-hop Delay

general.

Finally, we conclude that knobs and dials are features that applications can exploit to achieve greater resilience, both in their own application-specific communications and in supporting communications between other applications.

4 Information sources for resilient networking

For network resilience, cross-layering is seen as a support mechanism. The aim is to provide information about the nature of a challenge from the protocol layers for use by an adaptation component. Cross-layering is linked to measurement, because the provision of information about a challenge from the underlying protocol layers is basically a measurement task. The measurement system might choose the specific measurement mechanisms appropriate in certain use cases.

Usually, common cross-layering mechanisms are composed of static components, whose measurement capabilities are fixed. Active control, whereby measurement capability can be adapted dynamically, is not generally considered. For resilient networks, however, active components are essential for providing information about a challenge. The challenge might have dynamic and/or complex characteristics that have to be analysed in a number of steps meaning that several measurements may have to be taken. Basic research questions are thus: i) what to measure (i.e., what information has to be provided), and ii) where to deploy the measurement (i.e., what components have that information).

The resilience problem requires a broader perspective on cross-layering, because we need to obtain information about the nature of challenges to inform decisions on remediation. The information we need could be obtained externally (i.e., context information, as discussed in Section 2.3). Context can be provided to decision components to inform their decisions on detection, remediation, or recovery.

For this purpose, in the following subsection, we suggest three main requirements for the information sources required for resilience. We will use three case studies for the subsequent discussion: a BGP misconfiguration, a cross-layer design, and weather disruption in millimetre wireless mesh networks.

4.1 Information requirements

Let us investigate a simple decision in the case of BGP misconfiguration [Zmi09], in which a system would reason about whether to accept or drop routing updates. In this scenario, some routers that have identified a malformed update have dropped the session with every router that has sent that update to them. Other routers have accepted the propagated update. In other words, the routers have suddenly stopped functioning, whereas previously they were happily exchanging data. When this happens in a large number of routers, a network outage is experienced. If we return to our earlier questions: what information should be measured, and where should this information come from, to help avoid or minimise network disruption. If we address these questions, gathered information can be provided to an analysis component to make an effective decision on remediation (i.e., to accept or drop the specific routing updates which have caused the network instability). We might formulate several requirements for the information framework in order to tackle the stated problem.

I The information framework has to be able to select potential reasons for an *alert* by using a *knowledge base*.

   A monitoring system might probe end-to-end throughput. If a threshold is exceeded then an alert is generated. A number of potential reasons might be selected as to why an alert has been generated by using *resilience models*, such as end-to-end throughput oscillation models, that populate a knowledge base. Similar models for rate adaptive MAC protocols and jamming attacks could exist – end-to-end throughput oscillation can be experienced in both cases. However, the other models would be not relevant to the alerts. Therefore, the information framework first of all needs to select potential reasons for alerts, which can be used when performing further analysis. For the framework to be successful, it is important not to exclude the correct reason, if this can be discovered. This approach to selecting the cause of alerts using resilience models is synonymous with the multi-stage approach to challenge identification that is described in ResumeNet Deliverable 2.2b [SFM$^+$10].

II The information framework has to be able to *exclude* potential reasons by performing appropriate measurements.

   A large number of reasons might be selected as potentially relevant to a particular end-to-end throughput-oscillation problem. The model of each selected reason should be evaluated and checked to determine whether the end-to-end throughput oscillation could be caused by the corresponding reason, and if not, the reason can be removed, thus creating a short-list of potential reasons for the challenge. In the case study for rate-adaptive MAC, the information framework could check settings of the cross-layer system. The framework could evaluate the inner and outer loop of the cross-layer system. If the cross-layer system is set up to be robust in the particular environment, then the next model will be evaluated from the scope.

III The information framework has to be able to perform *more detailed* tests.

Several models could be excluded in the previous stage by performing certain measurements. The information framework might not be able to analyse the case further. The framework might not identify the information with the required level of certainty. Therefore, the next requirement for the framework is that it has to be able to perform more detailed tests in order to increase the certainty in the identification process. Other measurements or their combinations might be executed to support a decision.

The proposed requirements might form the basis for a multi-level information framework. The concept enables to process any kind of information from the information sources in order to improve the decision on challenge. It is crucial for network resilience to identify information that can affect particular decisions.

A similar approach has been suggested in the ECODE project [BKL+09]. The authors proposed an algorithm for automated classification of network traffic anomalies. Automated classification intends to add meaningful information to the alert of a detected anomaly. The authors aimed to show how the information obtained by further analysing the identified anomalous flows can be used in a signature-based classification module to reliably characterize different types of anomalies such as DDoS attacks. Two steps in the algorithm have been suggested. i, detect anomalies and identify related packets/flows. ii, use the packets/flows to obtain more measurements, and then apply signature-based classification of the anomaly to those measurements. It is advocated that these steps are necessary to obtain more information about the anomaly. The suggested algorithm processes captured packets/flows and makes further analysis over the collected data. However, this approach does not include the stage to perform more detailed tests when short-listed reasons might still not provide sufficient information about the context of a challenge.

In order to validate the concept, we can demonstrate its applicability to a number of case studies. Next, a categorisation of information that may be used by the information framework is presented.

4.2 Information categorisation

It might be argued that the most crucial ability of the information framework is the ability to perform more detailed measurements in order to identify information that is essential to make a remediation decision with a required level of certainty. Considering this requirement, information can be categorised in the following way, based on how it is derived: direct, indirect, related, full, or partial context. Direct context information is directly available in the system without need to change the structure of the system and its measurements. Indirect context information is located out of the scope of the system. Related context information is located out of the scope of the system with a relation to the direct context information. Full context information means complete information with a higher level of certainty than required. Partial context information means information with a level of certainty less then required. This categorisation might cover a broader variety of applications when we reason about information that is not normally available to certain decisions.

I **Direct versus indirect context information**

The direct context can be measured directly with the selected measurement mechanisms. For instance, the status of the channel where the jamming attack can be a potential reason which cause the end-to-end throughput oscillation. The indirect context might be rain.

The rain might cause disruption in the links of millimetre wireless mesh network. In order to identify the indirect context additional measurements have to be conduced.

II **Direct versus related context information**

The computer network has a distributed character. The dependability between components can play a certain role in the information analysis. The overall picture over the system could be helpful for performing further measurements on related context in order to identify essential information for the decision. The location of the information sources can be used as one of potential reasoning attributes. The information could be from the local sources, external sources, or network wide sources. For instance, the detection of selfish nodes in the opportunistic networks, where the system will look for related context will reason where to get information from, where to execute the search.

III **Full versus partial context information**

Full information about the cause could be obtained by the measurement. The system could conclude with the full certainty that this is the reason that caused the problem. In the knowledge base the rain model for millimetre wireless networks is populated. The system will check the weather from the weather forecast system for a particular geographical location. The granularity of the information is sufficient enough to make the decisions in the network to reroute the traffic around the affected regions. On the other hand, context information might be partial; this is because the disseminated information might be dropped or corrupted on the way to the recipient, or just the detailed information about the affected region might not be available from the weather forecast system to the required extent.

4.3 Application to case studies

A preliminary investigation of the information framework has been carried out in the following case studies. The first looks at a BGP misconfiguration case from February 2009 [Zmi09] causing a network outage. We chose this case study as it can be particularly hard to detect, such that an understanding of the problem can be clearly identified, and appropriate remediation can be taken. The second study looks at an end-to-end throughput case [KK05], where a poor cross-layer design or jamming attacks can cause degradation. This case study shows how the information framework could be used to improve a situation using multi-level information. In the last study, we look at how context information can be used for millimetre wireless mesh networks, where rain reduces the performance of the network [ABVJ08].

4.3.1 Case study 1: BGP misconfiguration case

An unusual routing update has been spreading over the Internet, and has caused a significant network outage. A number of routers have seen the update as malformed, while others have accepted the update. Those routers which have seen the update as malformed drop the session with all routers which have sent the update to them.

One question is what should be measured and where the measurement should be placed in order to provide essential information to a detection and remediation component for further analysis and processing.

With respect to the requirements stated for the information framework, the potential reasons need to be selected from the knowledge base of the resilient network. From the

Table 3: Summary of steps carried out in the information framework

Phases	BGP misconfiguration	A bad cross-layer design	Weather disruption
1. Selection phase	ports check, peers address check, AS number check, BGP misconfiguration error, network congestion, flapping, integrity mismatch, application reasons	dynamicity of control loops, selfishness of nodes	packet delivery ratio, error rate, weather status
2. Excluding phase	configuration error, network congestion, routing flapping	dynamicity of control loops	weather status
3.Phase for performing more detailed tests	history, unusual AS paths	frequency analysis of adaptation control loops	affected regions, the strength of rain in the regions, turbulences, validity of the rain information, number of hops to be foreseen and included in the calculation

populated BGP model, protocol states might be extracted for evaluation. BGP nodes could be in one of the following states: ‘idle’, ‘connect’, ‘active’, ‘open sent’, ‘open confirm’, or ‘establish’. There might be several reasons why the state cannot complete its functionality. In ‘idle’ or ‘connect’, it might be when TCP port 179 is for some reason closed, when TCP ports over 1023 are not open, or when the peer address or AS number on the router might be configured incorrectly. In the active state, it might be because of a BGP configuration error, network congestion, or flapping network interfaces. In the open sent state, it might be because of an integrity mismatch or other application reasons. Thus, when the framework would obtain potential reasons to the challenge, the framework should exclude potential reasons from the scope based on the results from performed measurements. For example, the ports might be checked by a port scanner. When the measurement would not confirm a particular problem with the ports, then it evaluates the next potential reason from the scope. The result of this excluding process might be a set of reasons (configuration error, network congestion, or routing flapping) which might still not provide the essential information for the further analysis for the decision. The third step would be to perform more detailed tests by trying to identify indirect or related context.

Monitoring sensors might detect that a single prefix caused an unusual high update rate. Some routers have sent the update further, others have dropped, and dropped the session with routers which have sent the update to them. Two routers have been exchanging data normally based on the evaluation of the logs, and suddenly stopped the communication. The alternative paths have been explored, former routes recovered. This event was experienced in thousands of other routers causing the routing stability problem. The sensors might detect a high increase of the rate in the routing updates. But this would still not give much information on the context what might be related to the cause of the routing instability. Another context information to be evaluated can be the routing history. The average AS path length is 4. There is no reason why the average length should be significantly larger than 4. If a particular update would start to cause the routing instabilities within a certain area, then the updates can be marked as suspicious, and the context information is identified. The context information might be used by a remediation component which will then reason only about manipulating with the length of period for the routing update being propagated further and thus decreasing the impact of the degradation.

4.3.2 Case study 2: A cross-layer design example

Consider a wireless communication example, in which end-to-end throughput degradation may be caused by a jamming attack, or by unsuitable cross-layer settings in the rate-adaptive MAC where data is sent at higher rates when the channel quality is good. A degradation could be caused by interferences on the links. An active measurement system would check the likelihood of degradation caused by such interference.

The authors [KK05] have pointed to a problem that might be experienced in a protocol design. They have showed that in certain circumstances end-to-end throughput can be degraded caused by the cross-layer design (rate-adaptive MAC) when the minimum hop routing is used. The goal of the rate adaptive MAC is to send data at higher rate when the channel is good by changing the modulation scheme. There are two control loops (inner, outer). The inner loop: each node controls its transmission power. The node increases the power when one hop neighbours is less then a control parameter. The node decreases the power when one hop neighbours is more then the control parameter. An outer loops controls the control parameter. It controls the control parameter based on the average end-to-end throughput. The outer loop works slower than the inner loop in order to avoid instabilities caused by the interferences. The question is what settings (what value of the control parameter) has to be chosen in order to avoid interferences and degraded throughput. The reasoning attribute is dynamicity of this cross-layer control.

This case study is about checking whether the challenge has been caused by jamming attacks on "bad" cross-layer settings in order to exclude the suspicious reasons selected from the knowledge base, by using an "active" measurement in order to provide the essential information for further resilience analysis.

So, a probe ("active" measurement) is sent to determine whether a measured performance degradation is caused by either a problem with the two control loops interacting poorly or a jamming attack.

We are improving the situation in a way that we have "excluded" suspicious reasons from the scope by using measurements in order to provide "essential" information to a decision component. The decision component now can reason about the cross-layer design only and does not need to consider jamming attacks.

In order words, when a system is using the knowledge base to select potential reasons to a challenge based on syndromes. Then we need to exclude the potential reasons from the scope by performing measurements in order to identify essential information of a challenge for further analysis.

4.3.3 Case study 3: Weather disruption in millimetre wireless mesh networks

Consider a millimetre wave wireless network, which can be badly disturbed by a large rain storm. Due to such an event, the error rate will significantly increase and the packet delivery ratio will drop. If the nodes have contextual information available on the rain intensities in the particular area (for example a suitable weather forecast), then the nodes might re-route effectively around affected regions and thus reduce the rain's impact on network performance [AJA+09].

The active monitoring system in that case needs to obtain the weather status information for a certain geographical area, and that information needs to be redistributed over the network. First, the context system might reason about the granularity of the available information. Second, it can reason about the search depth of the network from a particular node to effectively re-route affected regions. The depth might depend on the number of hops to be seen.

Another context process can evaluate whether the information is outdated or still current, in order to increase the efficiency of the remediation when the disseminated information has been corrupted and not available for further processing. The evaluation should consider the strength of the rain activities in the area, the speed of the rain, or the turbulences of the rain and thus determine the likelihood that this particular context information is still valid.

4.3.4 Summary

In the first case study, the related and indirect context has been explored by the context system. The system has evaluated the historical routing data after the excluding process has not provided sufficient information for the further analysis on the detection of the challenge. The context system has compared the AS paths to the average values. The context system has identified an update, which is significantly higher than the average and causing the sessions dropping in an area. The update has been marked as suspicious. Then this essential information has been provided to the remediation component which then instantiate an effective mitigation.

In the second case study, the dynamicity of the cross-layer design and selfishness of nodes is used for the search of context information causing the end-to-end throughput degradation. By conducting the "active" measurements the suspicious reasons are excluded from the scope of the context system and thus providing the information further for the remediation of the challenge.

In the third case study, the context information is formed by considering weather activities in a particular area. This context information is helpful for more effective re-routing around the affected regions, which has been reported with, for instance, high rain intensities, and thus decreasing the impact of the degradation.

5 Conclusion

The ability to provide information that can affect the decision of challenge mitigation is essential for network resilience, as it guides us on what to measure and where to measure. Key ingredients to tackle this problem are selection and deployment of measurement mechanisms. We have thus started to explore the information sources, suggesting the use of cross-layering for getting information across protocol layers, distributed network monitoring for getting a view from different angles on the network, and context for getting essential information from other, appropriate sources of information to assist the process of challenge identification and mitigation.

We have presented a formalism for cross-layer information sharing and control that allows us to consider the trade-offs that can be made at different protocol levels. We have identified the choices of end-to-end and hop-by-hop error control as a specific cross-layer scenario for further study of cross-layering, where applications and routers might improve each other's performance by exchanging information normally hidden behind layer abstractions.

By merging the three views on information (cross-layering, monitoring and context), we have suggested three components for the implementation of an information framework to realise the $D^2R^2 + DR$ resilience strategy. It has to be able to select for analysis an initial set of potential reasons for a challenge by using the knowledge base, reduce that set to a short-list by performing active measurement, and perform more detailed tests on the short-listed reasons to isolate the true cause.

The applicability of the suggested requirements for the information framework has been demonstrated by application to the case studies. We have looked at a BGP-misconfiguration incident causing the significant network outage, an example of cross-layer design causing end-to-end throughput degradation, and rain disruption to the performance of a millimetre wireless mesh network.

Future work in the area of cross-layer optimization and multi-layer resilience should focus on developing the information framework with a view to informing the ResumeNet implementation strategy.

References

[AA09] Ashok Anand and Aditya Akella. Netreplay: a new network primitive. *SIGMETRICS Perform. Eval. Rev.*, 37(3):14–19, 2009.

[ABVJ08] A.Jabbar, B.Raman, V.S.Frost, and J.P.G.Sterbenz. Weather disruption-tolerant self-optimising millimeter mesh networks. *Third International IFIP/IEEE Workshop on Self-Organizing Systems*, pages 242–255, December 2008.

[ADB+99] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[AJA+09] A.Jabbar, J.P.Rohrer, A.Oberthaler, E.K.Çetinkaya, V.S.Frost, and J.P.G.Sterbenz. Performance comparison of weather disruption-tolerant cross-layer routing algorithms. *28th IEEE Conference on Computer Communications*, pages 1143–1151, April 2009.

[Alj03] Hassan Aljifri. Ip traceback: A new denial-of-service deterrent? *IEEE Security and Privacy*, 1(3):24–31, 2003.

[BBC97] P. J. Brown, J. D. Bovey, and X. Chen. Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64, October 1997.

[BDIM04] Paul Barham, Austin Donnelly, Rebecca Isaacs, and Richard Mortier. Using magpie for request extraction and workload modelling. In *OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation*, pages 18–18, Berkeley, CA, USA, 2004. USENIX Association.

[Bis08] A. Biskop. Context services in content distribution networks. Master's thesis, Universitat Mannheim, February 2008.

[BKL$^+$09] Chadi Barakat, Amir Krifa, Yann Labit, Imed Lassoued, Johan Mazel, Philippe Owezarski, and Kavé Salamatian. Implementation of adaptive traffic sampling and management, path performance monitoring and cooperative intrusion and attack/anomaly detection techniques. Deliverable 3.2, The ECODE Project, October 2009.

[CEC05] Anupam Chanda, Khaled Elmeleegy, and Alan L.and Zwaenepoel Willy Cox. Causeway: Support for Controlling and Analyzing the Execution of Web-Accessible Applications. In *Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference*, 2005.

[CKF$^+$02] Mike Y. Chen, Emre Kiciman, Eugene Fratkin, Armando Fox, O Fox, and Eric Brewer. Pinpoint: Problem determination in large, dynamic internet services. In *In Proc. 2002 Intl. Conf. on Dependable Systems and Networks*, pages 595–604, 2002.

[CKN$^+$00] Ch.Efstratiou, K.Cheverst, N.Davies, A.Friday, and L.Yr. Architectural requirements for the effective support of adaptive mobile applications. *Proceedings of Int Conf on MDM2001*, 2000.

[Def06] K. E. Defrawy. Proposal for a cross-layer coordination framework for next generation wireless systems. *ACM International Conference on Communications and Mobile Compting*, pages 141–146, 2006.

[DMF08] D.Kliazovich, M.Devetsikiotis, and F.Granelli. Formal methods in cross layer modeling and optimization of wireless networks: State of the art and future directions. *Heterogeneous Next Generation Networking: Innovations and Platforms, IDEA Group Inc.*, 2008.

[FF98] D. Franklin and J. Flachsbart. All Gadget and No Representation Makes Jack a Dull Environment. In *AAAI Spring Symposium on Intelligent Environments*, number AAAI TR SS- 98-02, 1998.

[Fis10] Matthias Fischaleck. Discovery of malicious nodes in p2p-overlay-networks using the X-Trace framework. Diploma thesis, Faculty of Informatics and Mathematics, Computer Networks and Computer Communication, University of Passau, Germany, 2010.

[FM07] F. Fu and M.V.Schaar. A new theoretic foundation for cross-layer optimization. *Technique report*, 2007.

[FMD98] F.P.Kelly, A. Maulloo, and D.Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *J. Operations Res. Soc.*, 49(3):237–252, March 1998.

[FPK$^+$07] Rodrigo Fonseca, George Porter, Randy H. Katz, Scott Shenker, and Ion Stoica. X-Trace: A pervasive network tracing framework. In *4th USENIX Symposium on Networked Systems Design & Implementation*, pages 271–284, 2007.

[GG01] G.Xylomenos and G.C.Polyzos. Quality of service support over multi-service wireless interent links. *Comp. Network*, 37(5):601–615, 2001.

[GM01] R. Gold and C. Mascolo. Use of context-awareness in mobile peer-to-peer networks. *IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 142–147, 2001.

[GSR06] G.Thamilarasu, S.Mishra, and R.Sridhar. A cross-layer approach to detect jamming attacks in wireless ad hoc networks. *IEEE Military Communications Conference*, pages 1–7, October 2006.

[HNBR97] R. Hull, P. Neaves, and J. Bedford-Roberts. Towards situated computing. In *First International Symposium on Wearable Computers*, pages 146–153, October 1997.

[JJM10] J.He, J.Rexford, and M.Chiang. Design for optimizability: Traffic management of a future internet. *Computer Communications and Networks*, pages 3–18, 2010.

[Joh98] Mark W. Johnson. Monitoring and diagnosing application response time with arm. In *SMW '98: Proceedings of the IEEE Third International Workshop on Systems Management*, page 4, Washington, DC, USA, 1998. IEEE Computer Society.

[JP04] J.Y.Boudec and P.Thiran. Network calculus: A theory of deterministic queuing systems for the internet. *Online Version of the Book Springer Verlag - LNCS 2050*, May 2004.

[JTWW05] J.McNair, T.Tugcu, W.Wang, and W.Xie. A survey of cross-layer performance enhancements for mobile ip networks. *Computer Networks*, 49:119–146, October 2005.

[JWPA06] J.R.Goodall, W.G.Lutters, P.Rheingans, and A.Komlodi. Focusing on context in network traffic analysis. *IEEE Computer Graphics and Applications*, 26(2), 2006.

[KK05] V. Kawadia and P. R. Kumar. A Cautionary Perspective on Cross-Layer Design. *IEEE Wireless Communication*, pages 3–11, February 2005.

[KSK02] K.Chen, S.H.Shah, and K.Nahrstedt. Cross-layer design for data accessibility in mobile adhoc networks. *Wireless Personal Communications*, 21:49–75, 2002.

[Kum85] P. R. Kumar. A survey of some results in stochastic adaptive control. *SIAM J.Control and Optimization*, 23(3):329–80, 1985.

[LC90] J. D. FEDOR M. SCHOFFSTALL M. L. and DAVIN C. CASE. Simple network management protocol (snmp). may 1990.

[LUO02] L.Larzon, U.Bodin, and O.Schelen. Hints and notifications. *Conerence, IEEE Wireless Commun. and Network*, March 2002.

[LWC03] L.Capra, W.Emmerich, and C.Mascolo. Carisma: Context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering*, pages 929–945, 2003.

[MA02] M.Khedr and A.Karmouch. Acan- ad hoc context aware network. *IEEE CCECE'02*, pages 12–15, 2002.

[Mal93] G. Malkin. Traceroute Using an IP Option. RFC 1393, January 1993.

[MLPY07] M.Wang, L.Ci, P.Zhan, and Y.Xu. Applying wireless sensor networks to context-awareness in ubiquitous learning. *International Conference on Natural Computation*, 5:791–795, 2007.

[MMD07] M.Sifalakis, M.Fry, and D.Hutchison. *A Common Architecture for Cross Layer and Network Context Awareness*, volume Volume 4725/2007 of *Lecture Notes in Computer Science*. 2007.

[MSCJ06] M.Chiang, S.H.Low, A.R. Calderbank, and J.C.Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of IEEE*, December 2006.

[MSP07] M.A.Razzaque, S.Dobson, and P.Nixon. Cross-layer architectures for autonomic communications. *Journal of Network and Systems Management*, 15:13–27, 2007.

[NM07] N.Baldo and M.Zorzi. Fuzzy logic for cross-layer optimization in cognitive radio networks. *Consumer Communications and Networking Conference*, pages 1128–1133, January 2007.

[Pet09] Stefan Peters. Identifizierung von Routingfehlverhalten in IPv4 Netzwerken mit Hilfe des X-Trace Frameworks, 2009.

[PL04] A. Papachristodoulou and J.C. Doyle L. Li. Methodological frameworks for large-scale network analysis and design. *ACM SIGCOMM Computer Communication*, 34:7–20, July 2004.

[RJPS08] Justin P. Rohrer, Abdul Jabbar, Erik Perrins, and James P.G. Sterbenz. Cross-layer architectural framework for highly-mobile multihop airborne telemetry networks. *Proceedings of IEEE Military Communications Conference*, 2008.

[RKW$^+$06] Patrick Reynolds, Charles Killian, Janet L. Wiener, Jeffrey C. Mogul, Mehul A. Shah, and Amin Vahdat. Pip: detecting the unexpected in distributed systems. In *NSDI'06: Proceedings of the 3rd conference on Networked Systems Design & Implementation*, pages 9–9, Berkeley, CA, USA, 2006. USENIX Association.

[RPM98] N. S. Ryan, J. Pascoe, and D. R. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications in Archaeology 1997*, British Archaeological Reports, Oxford, October 1998. Tempus Reparatum.

[RS04] John Reumann and Kang G. Shin. Stateful distributed interposition. *ACM Trans. Comput. Syst.*, 22(1):1–48, 2004.

[SAW94] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, December 1994.

[SDE+10] 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*, 54(8):1245–1265, June 2010.

[SFM+10] P. Smith, M. Fry, S. Martin, L. Chiarello, M. Fischer, C. Rohner, G. Popa, H. De Meer, A. Fischer, and N. Bohra. New challenge detection approaches. Deliverable D2.2b, The ResumeNet Project, August 2010.

[SM05] V. Srivastava and M. Motani. Cross-layer design: a survey and the road ahead. *IEEE Communications Magazine*, 43(12):112–119, 2005.

[SPS+01] Alex Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-based ip traceback. pages 3–14, 2001.

[STP03] S.Shakkottai, T.S.Rappaport, and PC.Karlsson. Cross-layer design for wireless networks. *IEEE Communication Magazine*, 41(10):74–80, October 2003.

[SWKA00] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, 2000.

[WMS01] W.Zhang, M.S.Branicky, and S.M.Phillips. Stability of networked control systems. *IEEE Control Systems Magazine*, January 2001.

[WR03] Q. Wang and M. A.A. Rgheff. Cross-layer signalling for next generation wireless systems. *IEEE Wireless Communication and Networking*, pages 1084–1089, 2003.

[XXHL05] X.Gu, X.Fu, H.Tshofenig, and L.Wolf. Towards self-optimizing protocol stack for autonomic communication. *2nd IFIP International Workshop on Au- tonomic Communication, Springer Lecture Notes in Computer Science*, 3854:183–201, October 2005.

[YC08] Y.B.Reddy and C.Bullmaster. Cross-layer design in wireless cognitive networks. *Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 462–467, 2008.

[Zmi09] Earl Zmijewski. Reckless driving on the internet. *Renesys blog*, 2009.