ANA Project

Autonomic Network Architecture

Sixth Framework Programme
Priority FP6-2004-IST-4
Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D.3.6v1

The resilient part of the ANA architecture (v1)

- Towards Resilient Systems -

Version 1.0

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP0/D0.0
Document Title	The resilient part of the ANA architecture (v1) - Towards Resilient Systems -
Workpackage	WP3
Editor	Dimitrios Pezaros (ULanc)
Authors	Dimitrios Pezaros (ULanc) Angelos Marnerides (ULanc) David Hutchison (ULanc)
Reviewers	Guy Leduc
Dissemination level	Public
Contractual delivery date	31st December 2007
Delivery Date	12th February 2008
Version	1.0

Abstract:

This document presents the design and evaluation of two distinct yet complementary Resilient Systems that implement diverse mechanisms of the ANA Resilience Framework.

Keywords:

Resilient Systems, Networks, Measurement, Monitoring, Control, Cross-layer, Traffic Engineering.

Executive Summary

Previous work within the Resilience task of the ANA project has resulted in the definition of the six-step two-phase D^2R^2+DR strategy for resilient networked systems.

Although absolute resilience would in theory be desirable and could be achieved by adding resilient mechanisms at every communication layer and plane, in practice it will always be determined by finite resource trade-offs. Practical systems will be deployed when and where required to enhance certain resilience properties of networked systems, subject to resource availability and case-specific requirements. For example, network and content providers will deploy particular systems to protect specific resources which are subject to operational (e.g. traffic) anomalies. At the same time, the provision of predictable and optimised services can be achieved by diverse systems through continuous traffic monitoring and adaptive routing based on application-specific performance requirements.

This document describes the design of self-contained mechanisms to enable resilient communication by partially implementing principles of the Defend, Detect, Remediate, Recover + Diagnose, Refine (D^2R^2+DR) framework. We envisage that separate systems will undertake diverse measurement, monitoring, and control tasks that will target at optimising (inter-)network operation under certain challenges and conditions, and will have resilience as their main emergent property.

We demonstrate by example the two concentric modes of operation employed by diverse resilient systems, namely the real-time reactive control, and the always-on management and optimisation.

First, a cross-layer mechanism is described that employs flash-crowd detection and traffic engineering principles to defend and remediate network operation at the onset of bursty overlay traffic [55]. We have employed a trigonometric approach to detect sudden increases in request rate for a particular resource, and we have then evaluated several cross-layer application-network algorithms for traffic engineering that optimise global network resource usage based on compartment/ network -local knowledge.

And second, a measurement and control system is presented that performs adaptive routing based on temporal traffic performance levels [59]. Always-on measurement of diverse performance metrics experienced by the actual application flows is conducted, and then traffic is routed over paths/links that optimise a given performance aspect, according to application sensitivities. The particular applicability of this resilience mechanism in situations where no network-internal knowledge is provided for the underlying infrastructure is demonstrated.

Table of Contents

1 INTRODUCTION

The six step, two-phase D^2R^2+DR strategy for resilient networked systems that has been described in [68], provides a comprehensive framework to holistically optimise operational performance under the inevitable presence of faults, and adverse events and conditions. The framework considers two distinct and concentric modes of operation, a real-time control loop, and a background diagnosis and refinement plane. The former ensures that the networked systems' functional integrity is maintained in the presence of legitimate and malicious operational challenges by encompassing four distinct principal actions (D^2R^2):

- **Detection** of the occurrence of adverse events and conditions
- **Defence** against challenges and threats to normal operation
- **Remediation** of the effects of the adverse events or conditions to minimise their impact on the systems' operational state
- **Recovery** to the original operational state of the system

The latter ensures the operational optimisation of the networked system through the always-on activity of management and control mechanisms (DR) that:

- **Diagnose** sub-optimal system operation
- **Refine** behaviour based on measurement and on past (D^2R^2) cycles

The resilience framework has been represented by the cube shown in Figure 1, which depicts the three dimensions of multilevel resilience along with the resilience strategy dimension. The vertical axis show multilevel resilience by protocol layer and the horizontal axis show multilevel resilience with respect to the data, control, and management planes, along with cross-plane control loops. Layer-local functional composition and cross-layer optimisations are evident in the figure throughout the protocol stack.

Absolute resilience at any given layer is neither possible nor practical since the deployment of resilience mechanisms is determined by finite resource trade-offs. Processing, memory, bandwidth, power, latency and cost are factors influencing the relative composition and placement of mechanisms to ensure the overall networked system resilience, availability and optimal performance. Hence, although resilience mechanisms can be in theory deployed at individual layers and/or planes, resilience should be mainly viewed as a networked system *emergent* property that rises through the appropriate composition of measurement, monitoring and control (sub)systems and their cross-layer, cross-plane synergy. Moreover, it is realistic that resilient mechanisms will be deployed in response to (or in order to prevent) certain operational *failures*, and *abnormal network conditions*.

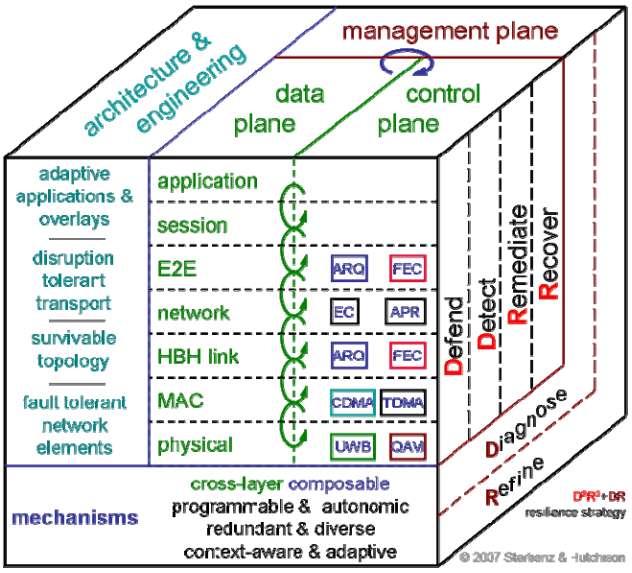

Figure 1: The Resilience Cube

Therefore, practical resilient systems should be expected to address certain *challenges* and *threats* to normal and optimal network operation, as opposed to being purely architecture-driven and complete with respect to some abstract model. This can be achieved be employing diverse and modular systems to implement certain subsets of the framework in order to increase network resilience under certain operational conditions. For example, different systems can undertake diagnosis and refinement tasks such as always-on monitoring and measurement of network performance, and real-time D^2R^2 tasks such as cross-layer traffic engineering to recover optimal network operation at the onset of legitimate traffic bursts.

In this deliverable, we describe the design and deployment of distinct systems that operate at the two different control loops of the resilience framework, and employ a subset of the identified D^2R^2+DR activities to provide for resilient communications under diverse operational conditions.

1.1 Scope of Deliverable

The description of practical *Resilient Systems* serves two major roles in the ANA project. First, it maps the guidance and architectural framework for network resilience to actual systems operating over a variety of network topologies, and taking control at the onset of diverse operational scenarios. Second, these self-contained systems that optimise network performance either in response to adverse operational conditions or as background always-on entities, demonstrate autonomic properties of systems that can perform feedback-based control and optimisation without requiring manual intervention or tuning.

We have purposely chosen to demonstrate and evaluate our two practical Resilient Systems over legacy networked infrastructures (Internet-based) in order to stress the general applicability of such mechanisms on existing networked systems. However, we trust they would fit the same purpose and adhere to the same operational principles if

deployed over a new ANA (inter-)network that would adhere to the notion of network compartments for point-to-point and end-to-end connectivity. Moreover, the rich service composition and clustering capabilities accounted for in the ANA framework (blueprint) would enable for even richer resilience services/systems to be deployed that would not be limited by the strict Internet-based layering and inter-domain rules.

1.2 Structure of the document

Following this introduction, Section 2 presents a real-time control loop system that **Detects** the onset of legitimate yet abnormal bursty requests, and then it employs cross-layer information exchange to **Defend**, **Remediate** and **Recover** the effects of such adverse (flash) events. Section 3 describes an always-on background **Diagnosis** and **Refinement** system that continuously monitors numerous service quality metrics of application flows over virtual links and then optimises perceived performance through flow-based adaptive routing.

2 EXPLICIT CROSS-LAYER OPTIMISATION

Cross-Layer optimisation relies on the relationship and exchange of metrics information between an overlay and an underlay network. Overlay networks, place commitment and trust only on application-level decisions [55] and have minimal knowledge of the underlying network structures. Explicit synergy between overlay topologies and the underlying infrastructure can enable both layers to interact through information exchange, and collaboratively alleviate sub-optimal operational conditions that affect performance and general service quality. Cross-layer optimisation is an approach that targets a better network performance and at the same time satisfies the needs for achieving a resilient network via providing features such as remediation, recovery and detection.

Through appropriate use of such mechanism, we can achieve the reliable detection of traffic anomalies that at the ingress points of ISP topologies, and subsequently the defence and remediation of their impact on the overall network traffic and global topology. In general, the proposed approach fits well the objectives of a real-time control loop and in parallel sets a strong basis for a background diagnosis and refinement mechanism as defined by the six-step two-phase resilience strategy in [68].

This chapter is structured as follows. Section 2.1 introduces the exact problem and specifies some of the problem's expected consequences. It also provides a clear motivation for remediation and recovery through the explicit cross-layer information exchange. Section 2.2 concentrates on the specific case of Flash Events (FE)s in Client-Server (CS) and overlay systems, and an approach for detecting their onset. Section 2.3 discusses the algorithms for remediation and recovery, and demonstrates how local (network/compartment-wide) knowledge can provide for simultaneous global optimisation of metrics at the application and network layers.

2.1 Problem statement

Overlay networks surely hold large popularity among users in the Internet and their applications provide various services such as file sharing, application-level multicast, scalable object location and network-embedded storage. Nevertheless, such applications mostly take implicit or explicit control over the routing that traditionally takes place at the underlay network layer [56]. Control over routing is mostly based upon requirements settled by each application belonging to a specific overlay network (i.e. Bit torrent), such as end-to-end delay, loss rate or throughput [36].

This application-layer routing control made on any overlay is possible to lead to sudden changes, flaws and unpredictable traffic dynamics over the underlying network

infrastructure. Any application or overlay network in particular take an underlay ISP as a purely blackbox component and takes any routing decision mostly in independent manners in a way to neglect any requirements or policies coming from the underlay. It is acknowledged that backbone network infrastructures are engineered in a fashion of considering matrices of traffic aggregates on large time scales that are very coarse-grained in comparison with the reaction times of overlay networks [55].

Further, by having multiple overlays operating simultaneously over areas of the Internet in their own independent and dynamic manners, there is absolutely a threat for any ISP to go under load-balancing and traffic engineering instabilities, and henceforth a cost-ineffective situation.

The problem we address here has been analyzed by others and in [48] there is a suggestion of exposing global topology and/or performance information to overlay networks when they need to take their application-specific routing decisions. In theory, such approach could be beneficial for the end-to-end operation of diverse application-level structures and topologies; however it may not be realistically feasible due to issues of global scale and also the fact that ISPs would never expose local traffic or topology information, since these are considered trade secrets and competitive advantage sensitive.

Our approach is therefore based on the idea that both underlay and overlay layers should synergistically co-exist, and at the same time assume that there is no global information exposed. The required information would be exploited locally by the underlay infrastructure within the realms of an administrative domain and provide services to the overlay. Furthermore the underlay will try to optimize the operations on both networks as well as its own resource usage. This exchange of information and right coexistence by both networks will facilitate correct traffic engineering methodologies and will also prevent any additional probe traffic being generated. In parallel, it will set a good basis for employing a detection mechanism for any abnormal events in traffic, taking place on either the overlay or the underlay layer.

We have examined and analysed the operation of overlay and underlay topologies at the onset of variable and bursty requests. We have selected to study the generic notion of an unstructured Peer-to-Peer (P2P) file sharing system due to the unpredictable traffic dynamics and the intensive bandwidth operation characterizing it. Also, due to their popularity, such systems often face a very high number or requests by numerous clients in a short time interval due to a distribution of a popular file. Under such conditions, their distributed servers may get virtually unreachable, resulting in the content not spreading fast to diverse segments of the network, and the stress put by the increasing number of requests not being relaxed.

The following figure presents our overall analysis scenario and in addition it shows how each mechanism we recommend satisfies and fulfils the requirements coming from the resilience framework (cube).

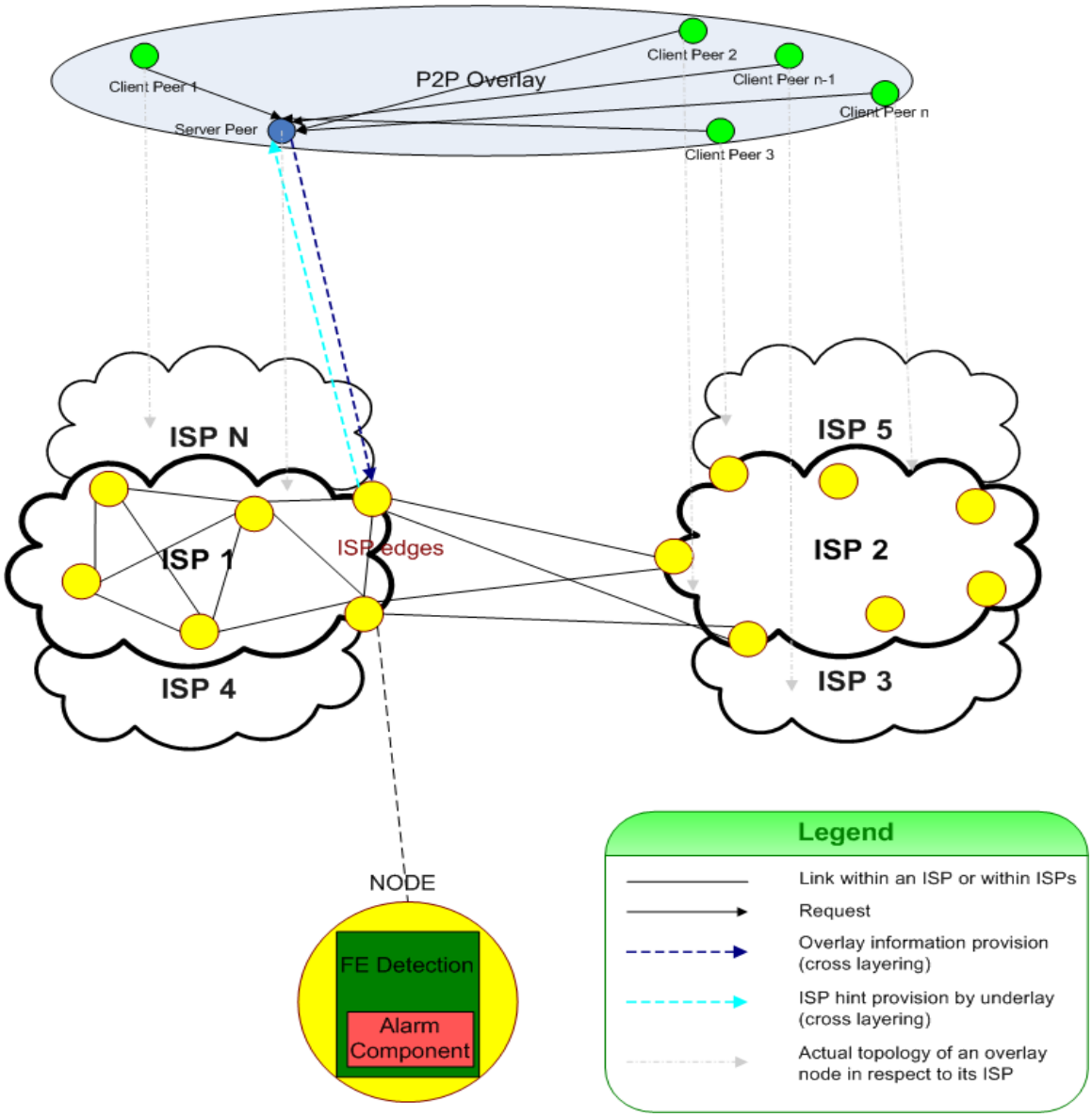

Figure 2: The overlay-underlay scenario and the resilient mechanisms

The two most basic components that support our needs for a resilient behaviour are the Cross-Layering and the FE Detection components. The Alarm Component is meant to be a sub-component of the FE Detection. FE Detection is based upon the general model for flash crowds and it's the core basis of our detect property in the resilient cube. Cross Layering is a mechanism that places a lot of importance in to the exchange of local network-level information (compartment/ISP hints) in order to geographically spread the content and further establish a defensive mode with remediation and recovery on the overlay. Thus, remediation and recovery may be achieved via a local network knowledge and global optimization. The following sections describe in detail the FE Detection, and the remediation and recovery mechanisms, respectively.

2.2 Flash Event's Detection

Several studies were made in the past regarding the nature and behaviour of Flash Events (FEs) (also known as flash crowds) [3]. A FE can occur on an underlay or an overlay network (i.e. P2P network) where in both cases the effects are still the same and sometimes related (i.e. a FE on an overlay that causes the content not to spread to geographically diverse areas of the network, also causes the link stress put on the underlay topology not to be relaxed). A *FE is the phenomenon where a server becomes virtually unreachable due to the multiple and numerous requests produced by various clients in a network.* All the requests in such a phenomenon demand the same data object on a server where that specific server is the only one with the possession of that specific data object. There are many times where such servers may be major news-webcast sites such as MSNBC [46], or other times may be unpopular sites that got famous after being mentioned in a popular news feed [29].

According to [29] a FE is decomposed in to three phases where each phase states the relationship between the requests (which can be referred to in a generalized form as traffic) arrived on a server within a certain time interval. The following figure shows these three phases in the event. Initially is the *ramp-up phase* having a (theoretically) linear increasing from a normal request rate $\mathbf{R}_{normal}$ to the maximum rate of requests appeared $\mathbf{R}_{flash}$ in a small amount of time $[t_0, t_1]$. Subsequently, there is the *sustained traffic phase* at $\mathbf{R}_{flash}$ between t_1 and t_2. Finally, the traffic decreases and eventually returns back to $\mathbf{R}_{normal}$ in the time space between t_2 and t_3; this is referred to as the *ramp-down phase*.

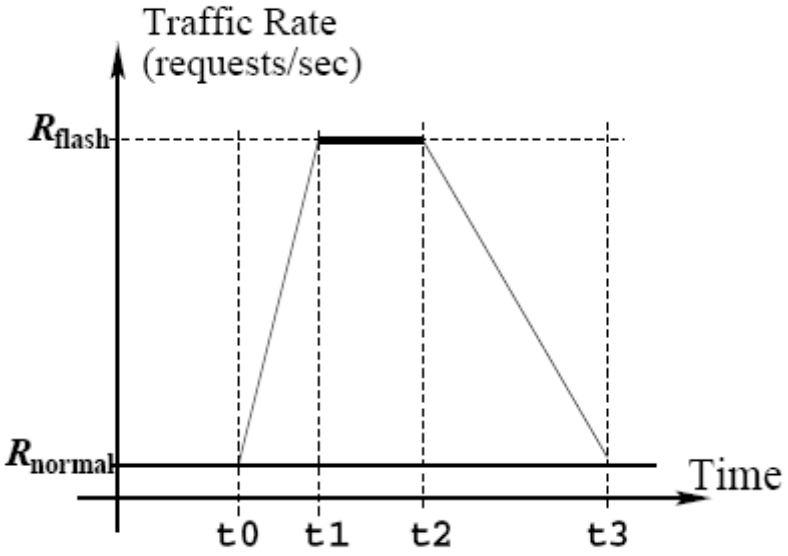

Figure 3: The general model for an FE.

2.2.1 The Flash Event detection mechanism

The unpredictability employed by a FE is depended upon issues such as the number of clients, heterogeneity (i.e. clients from different underlay or overlay networks) and further the exact nature of that event (i.e. expectable and normal increase on the number of requests). However, our proposed approach for detection mechanism relies upon the general FE model presented earlier and it should be functional on the ingress (i.e. gateway) points of an ISP network topology.

The approach taken in our detection mechanism is concentrated on the initial ramp-up phase of a FE where it will be possible to check on runtime if a sharp increase in traffic is taking place. As shown in the following figure, such an increase can be estimated with the use of the angle ***k***. The sharpness of the line is therefore the estimation coming from the value of the tangent on ***k***.

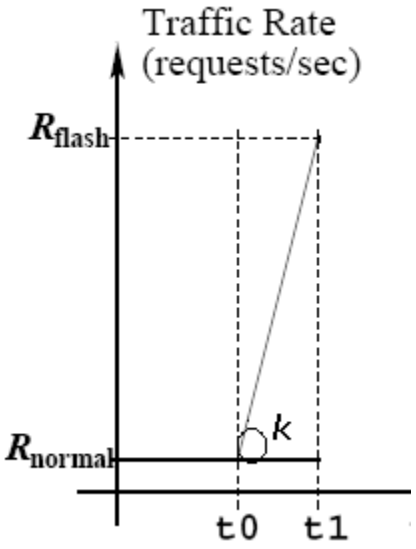

Figure 4: The ramp-up phase and the angle k.

Since:

$$tan(k) = (Rflash - Rnormal) / (t1-t0)$$

Under both theory and practise, ***k*** should be less than 90 degrees and greater than 0 (***0*** < ***k*** < ***90***). The ***k*** angle should be less than 90 since the increase in request rate cannot occur in 0-time Also, ***k*** should be greater than 0 due to the fact that we are dealing with an increase. However, in reality there should be certain values representing the time and the requests determining the acceptable values for ***tan(k)***. By setting a range of values coming from several known FE traces it would be easy to establish a prediction mechanism employing FE detection on runtime.

2.2.2 Real traces and the k angle threshold

In order to support theory it was necessary to place an effort and employ a threshold to the values for the ***k*** angle with the use of real values coming from real FE traces. This document provides a sample of traces found on the play-along site [29]. The online play-along website offers broadcasting of television shows. [29] presents a FE made on play-along in the case where a popular television show was broadcast and numerous users tried to access it on a limited amount of time. Using these traces found on graphs and figures from [29] we show a possible and realistic threshold for ***k***.

Two basic graphs were produced in [29] that show the general form of the FE (Figure 5) and a more detailed zoomed graph showing the ramp-up phase (Figure 6), respectively.

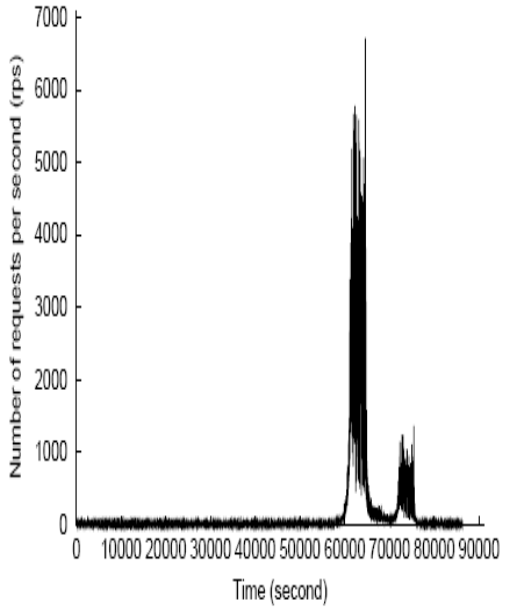

Figure 5 : The FE on play-along

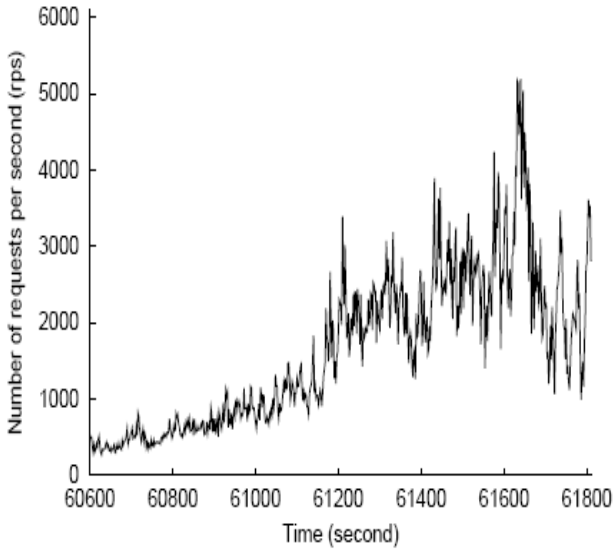

Figure 6 : The beginning of the ramp-up phase in the FE on play-along

A closer look to the above figures will provide a justification that in real scenarios the requests made while the ramp-up occurs are not strictly following a linear form but instead in some cases they can get a lower or higher value in small time periods. This poses a challenge regarding the assumptions we set in order to support the resulted threshold. For determining a ***k*** threshold we used the values from Figure 5 and we assume that dramatic increases on the number of requests in several time frames are important (i.e. dramatic increases on 61200 seconds or 61600 seconds) and they are not a scheme of a sporadic request increase event on the server; we also assume that the graph takes an increasing shape after 61800. The latter was assumed due to the shape of the graph

(Figure 5) showing the FE as a whole. As for the former, it should be kept in mind that sporadic increases may occur at any time on a server and they might not be a part of the ramp-up phase of an FE, however in this example we consider these as chained events constructing the ramp-up phase.

A core basis for our analysis was to follow a linear approach and determine a threshold; such an approach may be a good starting approach for a further and deeper investigation.

Initially we took the highest and lowest value of requests and then tried to form the given graph on the most linear shape (Figure 7); then the threshold estimation for the ***k*** angle's tangent was made using the formula showed previously (section 2.2.1).

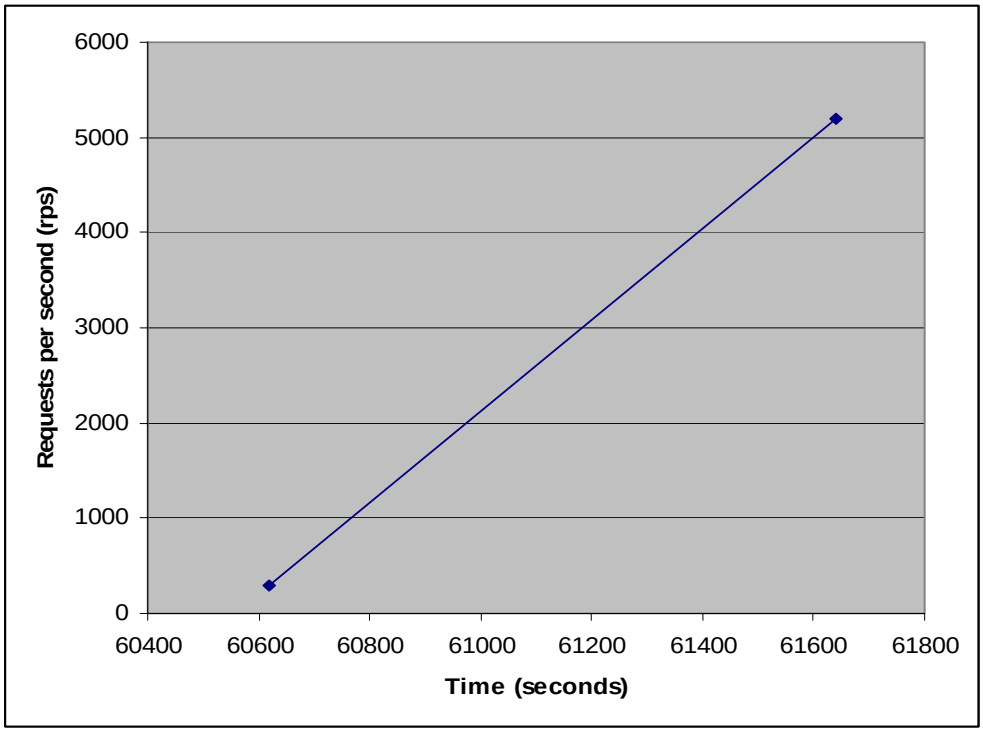

Figure 7 : The ramp-up phase on the most linear form using only the lowest and the highest values for requests per second

Therefore:

Highest (req/s) = 5200 req/s at 61640 seconds

Lowest (req/s) = 300 req/s at 60620 seconds

Based on those we can easily estimate the ***tan(k)*** variable and set the threshold

where: ***tan(k)* = (5200 – 300) req/s / (61640 – 60620) sec = 4.8**

Based on this result we have limits for the k angle and further the tangent of the k angle to: ***0< tan(k) =< 4.8***

With the use of this threshold and of course always under the assumptions made, we can use this value to employ a prediction mechanism for alarming a network that a FE might occur in runtime. One of the desired goals with the use of this ***tan(k)*** threshold-oriented approach is that of using it as the basis for an always-on mechanism on a single point in the network, and periodically check the values taken for ***tan(k)***. This periodical check will be depended upon request rates gathered on specified time intervals.

A requirement guaranteeing a reliable service from this detection mechanism tool would be that of specifying the exact time periods for the request rate collection. We recommend that the collection of data should remain on the seconds scaling and not minutes due to the fact that a FE's ramp-up phase is constructed and achieved in a small amount of time as justified on Figure 5. Furthermore, by looking at each time frame on that figure (200-second frames) we distinguish several increases/decreases on time periods less than 40 seconds. Based upon that, a data collection and triggering of the ***tan(k)*** threshold check by the proposed mechanism should be in a period of 20 seconds for preventing any loss of significant changes on the request rate.

The following section gives a more detailed description of the scenarios as well as the algorithms we used for a correct load-balancing where those compose a mechanism for remediation and recovery in the case where an FE takes place on an overlay network.

2.3 Remediation & recovery through local knowledge and global optimisation

As already mentioned in the previous section, a FE is caused by multiple requests from numerous clients requesting a famous content from a segment of an overlay (and consequently an underlay) network. A FE taking place on an overlay would easily affect the actual traffic flow on the underlay topology (i.e. a local ISP) causing at the same time issues of service delay to the several clients requesting (via their overlay application) the given data object.

We believe that the collection of local network-level information would be significantly important on spreading the content in right manners and remediating a possible FE on the overlay. We propose the concept of ISP hints as an elegant mean to provide application-network cross-layer optimisation [55]. ISP hints are obtained by an application through the explicit interaction with its access ISP, yet they do not contain any explicit information about the ISP's network structure or policies. The idea behind ISP hint services is very generic in the sense that the hint services are open and non prescriptive. An ISP could basically offer a range of such services, each taking specific inputs and returning specific hints. Applications would choose the hint service(s) that best fit their

needs.There can be various services offered as an ISP hint, however, we used two examples that satisfy our goals in manners of cross-layering.

The first such ISP hint service is called a “distance service”. It simply takes IP addresses as input and returns a distance measurement between each destination and the requester. The notion of distance can obviously be specified and measured in numerous ways, but again the advantage of using the ISP hints abstraction is that ISPs can choose which metric to provide and how to measure it, without even having to inform the applications (the requesters) of their choices. For instance, an ISP may decide to use Autonomous System (AS) path length as a distance measurement, while another may base its distance measurements on more complex embedded coordinate systems [50][51]. The “region aware clustering” is another example of ISP hint service. Here, the ISP would take a set of IP addresses as input, and return these addresses split into several subsets (clusters). In its simple version, all the addresses in the same subset/cluster would be reached, from this ISP, through the same egress border router. However, note again, that ISPs may want define “regions” in a different way. For example, the clusters in the simple region-aware clustering could be further split according to the second AS hop they would traverse, and so on and so forth. This latter version of the service could be called “2-level hierarchical region-aware clustering”. Note that the distance service based on AS path length, and both region-aware services described above can be implemented based solely on routing information available at an ISP.

Using these ISP hints, we have illustrated how applications can take advantage and improve their performance. This illustration was achieved with the use of two paradigms and our approach was mostly concerned with the content distribution taking place. The first example is based on the simple client-server (CS) paradigm where is very often used for file transfers (e.g. HTTP, FTP). In the simplest form of client-server transfers, the server receives requests from the clients which it serves. The server usually accepts new requests as long as it has enough local resources to accommodate them. On the other hand, an enhanced server could exploit ISP hints to increase the perceived service quality of its clients. Indeed, by using the simple region-aware clustering service, a server is capable of ensuring some form of load-balancing between its ISP egress points, thus controlling the contention that exists between its clients inside the network. To do so, the server could set a limit on the number of concurrent connections that it will open per cluster (i.e. ISP egress router), and use the simple region-aware clustering for admission control. More specifically, the server can use the ISP clustering service to decide which request to serve (if the corresponding connection uses an egress for which the connection limit, a.k.a. threshold, has not been reached) or queue. By limiting the number of concurrent downloads and ensuring load-balancing, the average download time should be reduced, improving user satisfaction. We have implemented two client-server (CS) algorithms and quantified the performance gains of employing ISP hints services. A traditional CS algorithm has been developed where one node serves simultaneous requests up to a certain threshold value *T*, on an aggregate First-Come-First- Served (FCFS) basis. Upon reaching this threshold, further incoming requests are queued and served as soon as existing transfers complete on a FCFS basis.

An enhanced CS algorithm has also been developed where the server uses ISP hints to perform load balancing on the incoming client requests. As with traditional CS case, up to a threshold T client requests are served simultaneously. However, in this case, the server uses the region-awareness ISP hint to cluster incoming requests based on the egress links response traffic is going to be routed through. Traffic is then load-balanced over the server's egress links. Three variants of this region-aware request clustering have been implemented. In the Simple Clustering (SC) variant, the server clusters requests to regions based on the first-hop egress link traversed by the response traffic. The total simultaneous request threshold T is divided by the servers' v egress links and $\tau = T / v$ simultaneous requests are served per-cluster. In the second-hop Flat Clustering (FC) variant, the server groups requests based on the second hop traversed by the response traffic, and performs load-balancing ignoring first hop information. Finally, in the second-hop Hierarchical Clustering (HC) variant, requests are clustered hierarchically based on the first and second hops traversed by response traffic. The initial threshold T is divided by the number of first-hop egress links v to produce i first-hop thresholds $\tau_i = T / v$, each of which is further divided by the number of egress links attached to first hop i. In all variants, when the region-aware clusters are computed, requests are served on an intra-cluster FCFS basis.

Our second paradigm is based upon a popular way to transfer files that of P2P file-sharing overlays. P2P file sharing comes in many flavours, so for this paper we chose to use a simplified version of a Bittorent-like distribution overlay as a reference [8]. A generic simple overlay algorithm has been implemented that includes two different peer entities, a central tracker of the content; and normal peers acting as both content clients and providers. Each peer registers with the tracker either as a provider or as a requester for a certain piece of content. In the former case, the tracker updates its list of providers, and in the latter case it acknowledges the client's registration by sending back the list of currently available providers for the specific content. If the list is too large, then a number of providers is chosen randomly and returned to the requester. The client randomly selects a provider from the list to request the content from. If it is denied service (because the provider has reached the maximum number of connections it will serve), it randomly selects a different provider from its list until either its request is accepted, or its providers' list is exhausted, in which case it times out and re-requests a providers' list from the tracker after a certain time interval. Upon successful completion of a download, a peer registers itself as a provider with the tracker.

We have developed a number of augmented and region-aware overlay algorithm variants to demonstrate how the operation of this simple overlay can be optimised using explicit cross-layer interaction through combinations of the two ISP hint services described in section II. The augmented overlay algorithm uses the ISP "region-aware clustering" hint (as in the CS case described above) for the provider peers to load-balance their response traffic. At the same time, a provider that has reached its simultaneous serving threshold explicitly redirects further clients to the subset of peers (providers) it has already served through the same "regional" cluster that the incoming request came from. This way, providers attempt to spread the overlay traffic load to diverse segments of the underlay (Internet) infrastructure. Requesting peers choose among alternative providers either

using a Random function (Ran), or by employing the ISP "distance" hint that returns an AS Proximity (ASP) metric and then selecting the least-AS-distant provider. Furthermore, a Region-aware Overlay (RegO) algorithm has been developed which implements Random (Ran) provider selection and region-aware load-balancing to requests.

In contrast to the augmented algorithm where clients are only redirected to a "region"-based subset of the alternative providing peers, RegO uses the central tracker entity that provides requesters with all currently providing overlay peers. All variations of the augmented and the RegO algorithms have been designed to implement first-hop (SC) and second-hop hierarchical (HC) region-aware clustering to load-balance competing requests. In the following section, we evaluate and quantify through simulation the performance gains of employing explicit cross-layer synergy for both the underlay and the overlay layers.

2.3.1 Simple client-server load balancing

We have used the Network Simulator (ns-2) [70] to assess the effect of the region-aware load-balancing for the client-server case over a variety of Internet-wide topologies. A piece of content hosted by a single peer on the topology is simultaneously fetched by a number of clients (assumed to be triggered by an external stimulus, such as central web-based advertisement). The server has a configurable threshold up to which it is willing to serve client peers. The source of the requested content resides behind a number of egress links which we varied in different experimental runs.

We have employed 103 client nodes using both symmetric and random node placement over Internet-wide Autonomous System (AS) topologies. Symmetric client placement assumes the same number of clients attached to each AS. Under random client placement, clients are randomly attached behind each AS using a multi-level number generator based on uniformly distributed random variables. We have conducted multiple simulation runs over different edge-degree topologies and using a varying threshold value of 5, 10 and 20 simultaneous transfers, to compare the mean individual transfer throughput between the region-aware load-balanced scenarios and the unbalanced aggregate First Come First Serve (FCFS) case.

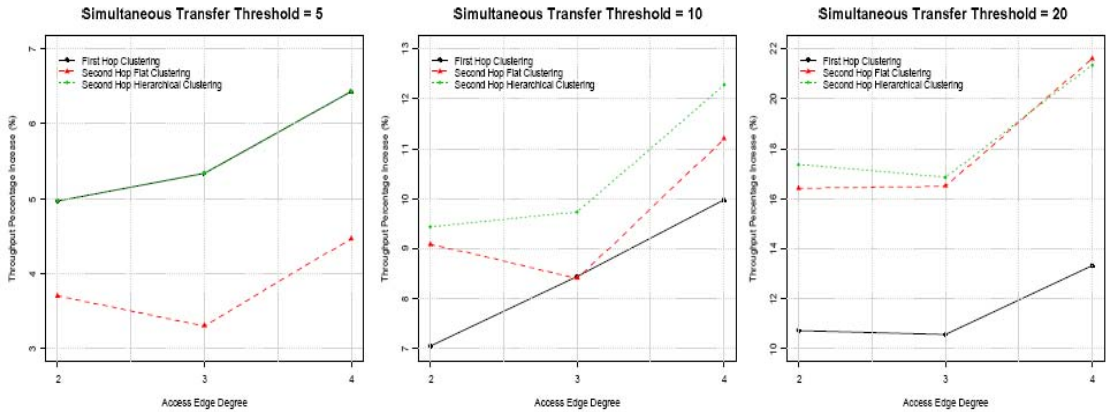

Figure 8 : Throughput percentage increase of load-balancing client requests under symmetric node placement for varying simultaneous transfer threshold values.

Figure 8 shows the percentage increase in throughput for the different simultaneous transfer threshold values over symmetric client node placement. Throughput has been measured as the number of bytes transferred between the server and the client over the duration of the flow and resembles the widely used Bulk Transfer Capacity (BTC) metric. The link bandwidths of each topology generated were uniformly distributed to accommodate for the randomness in the actual available bandwidths over the Internet due to the variable traffic dynamics of each segment. Figure 9 shows the percentage increase of transfer throughput over random client node placement using the same threshold values and topology-wide link bandwidth distributions. It is evident that significant increase in transfer throughput is achieved by the cluster-based load-balancing algorithms with respect to the unbalanced aggregate FCFS mode of operation.

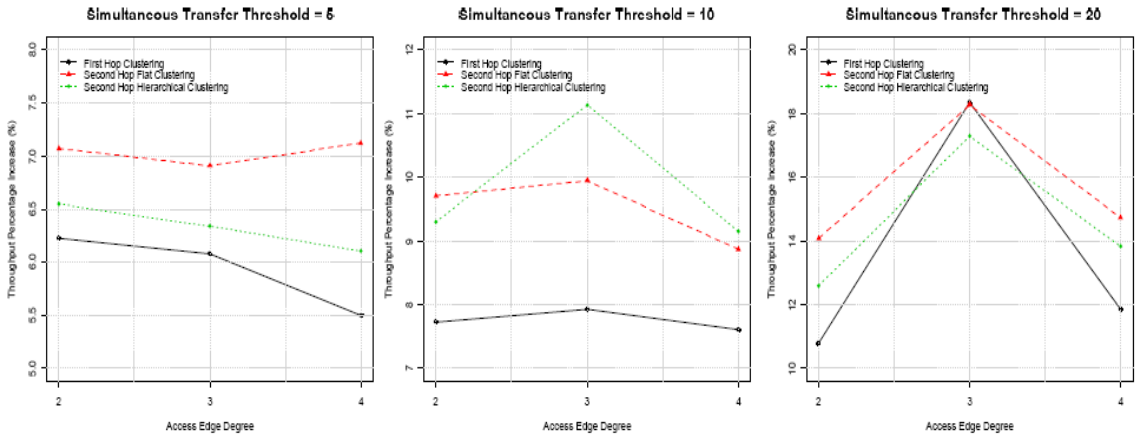

Figure 9 : Throughput percentage increase of load-balancing client requests under random node placement for varying simultaneous transfer threshold values.

Over symmetric client placement, even first-hop SC load-balancing can achieve a steady over 10% mean throughput increase as the simultaneous transfers and access edge degree grow larger. Load-balancing based on second-hop FC and HC achieve throughput gains of up to above 20%. First-hop (SC) and second-hop hierarchical clustering (HC) show a steady and proportionally increasing trend with the number of simultaneous transfers and access edge degrees. Second-hop Flat Clustering (FC) exhibits a less predictable performance gain due to equalising (in some cases) the per-egress (first-hop) load balancing, however, for increasing numbers of simultaneous transfers this algorithm also shows a steadily anodic throughput gain. As expected, over random client placement, the throughput gain is influenced by the uneven number of clients accessed through each egress link, and is henceforth less deterministic.

However, it is worth mentioning that throughput increase is still achieved by region-aware load-balancing in all cases, and also that absolute throughput values are in many cases larger than those of the corresponding threshold/edge degree values over symmetric client placement. Overall, it is evident that an ISP can spread popular content faster to diverse segments of the Internet and minimize the persistence of incoming requests by employing a simple load-balancing algorithm and information readily available within its BGP speakers. Finally, such a composed mechanism will surely help in the defined resilience framework in order to apply a right remediation and recovery strategy.

2.3.2 Region-aware Overlay algorithms

We have further investigated the gains of region-aware load balancing for complete P2P file-sharing overlays over representative AS-level Internet topologies. We have hypothesized that region-aware load balancing by every providing peer will help the content to spread quickly over diverse segments of the underlying infrastructure, and hence minimize the impact of persistent request and response traffic on a single ISP that (implicitly) hosts the popular content. At the same time, the ISP itself does not need to know anything about the internals of the content other than to identify which requests are for the same highly-popular object (through e.g. hashes), nor does it need to invest into infrastructural support for services such as caching to minimize the additional stress over its links.

We have evaluated the overlay/underlay interaction through comparative performance analysis of the different algorithms. Brite topology generator [11] was used to generate a large variety of representative topologies to include diverse topology models, AS node populations and minimum AS edge-degrees. We have generated a number of power-law AS-level topologies to include 100, 500, and 1000 nodes, each with a minimum edge degree of 2, 3 and 4 links per leaf AS [5][12]. Each simulation focused on both the initial bursty phase of the overlay when all peers simultaneously fetch a newly populated piece of content, and the steady-state phase of the system when content has spread among different peers.

The experiments assessed the overall performance gains of the explicit underlay/overlay synergy in spreading the so-called first chunk of content among the participating peers [60][61]. The chunk size was set to 1MB. A constant threshold of 10 simultaneous transfers has been used. The performance metrics measured for all the algorithms and their variants were the individual transfer throughput in KB/s, and the mean and maximum link

stress over the complete Internet-wide topology. As in the previous section, throughput is the BTC of each transfer, whereas mean and maximum link stress have been measured as the average and maximum number of flows active over each link of the topology, at any given time. The comparative results show that the region-aware algorithms improve both aspects of overlay and underlay performance.

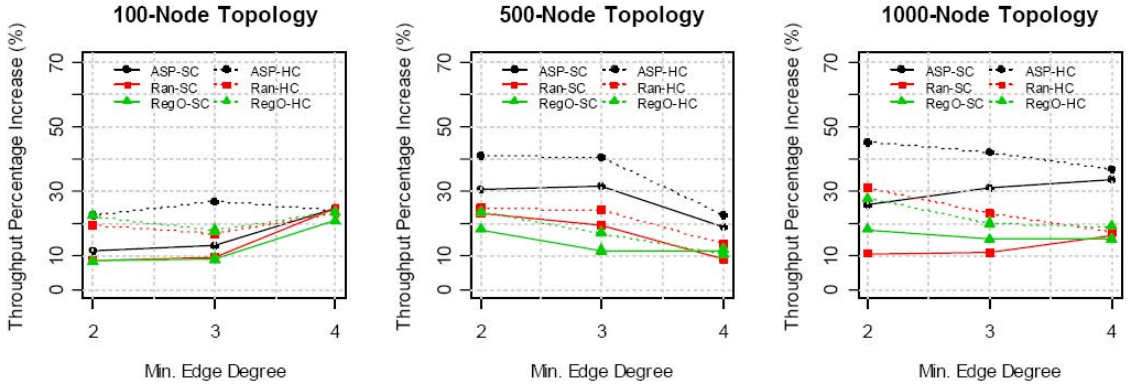

Figure 10 : Percentage increase in mean individual transfer throughput for the cross-layer algorithms

Figure 10 shows the percentage improvement in mean transfer throughput over different-size AS-level topologies, achieved by the cross-layer algorithms with respect to their pure overlay counterpart. The figure also shows the variations in throughput increase with respect to the minimum access edge degree of the topologies. Solid lines show the performance gains of the cross-layer algorithms with first-hop simple clustering (SC) and dashed lines show their second hop hierarchical clustering (HC) counterparts.

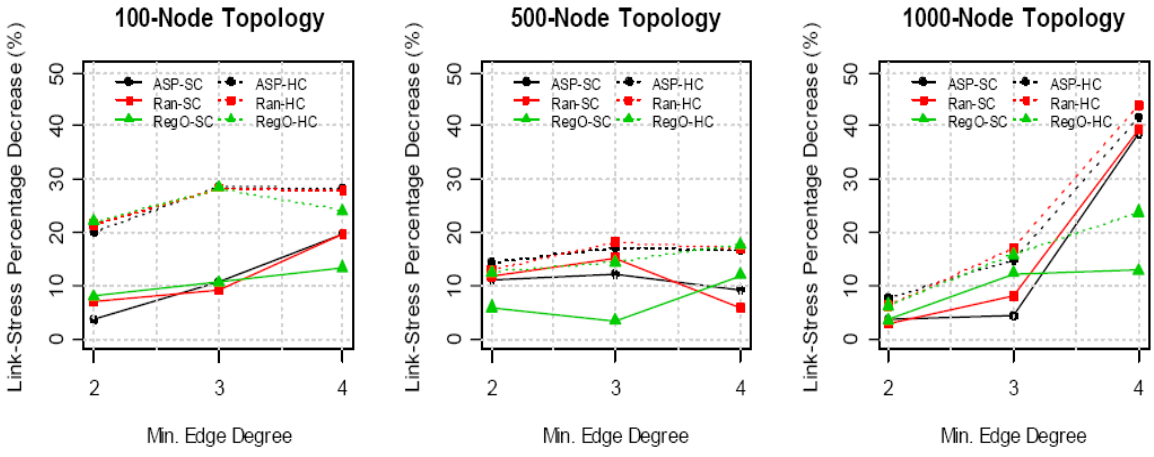

Figure 11 : Percentage decrease in topology-wide mean link stress for the cross-layer algorithms

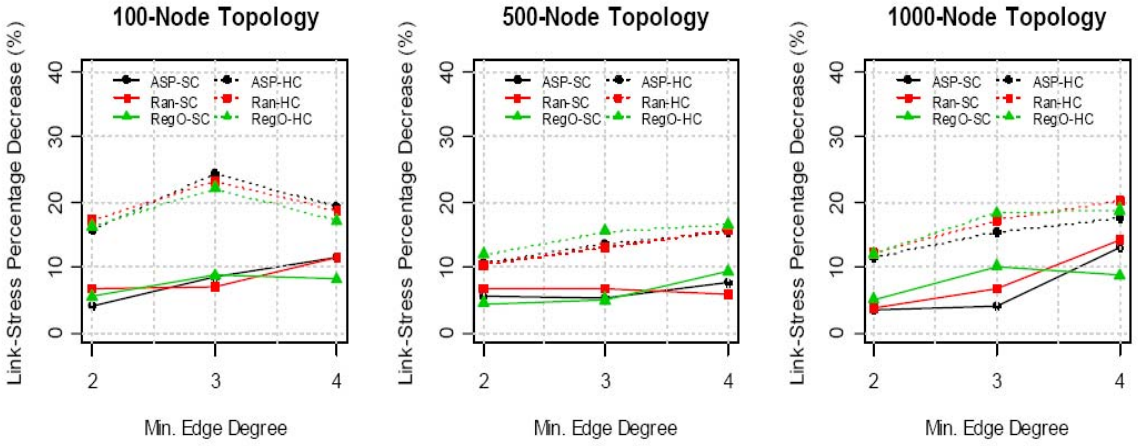

Figure 12 : Percentage decrease in topology-wide maximum link stress for the cross-layer algorithms

Likewise, Figure 11 and Figure 12 show the percentage improvement in the topology-wide mean and maximum link stress for the same set of AS-level topologies. Although there seem to be no clear correlation between throughput nor link stress, and the topology size and edge degree, it is evident that on average cross-layer algorithms outperform their simple overlay counterpart. For each algorithm, employing hierarchical clustering (HC) of the requesting peers consistently outperforms simple clustering based on network access link (SC). For mean transfer throughput, the augmented overlay algorithm with ASP provider selection exhibits significant gains over the rest of the algorithms, approaching 50% optimization. It is also worth noting that for the 1000-node topology with a minimum edge degree of 4, all variants of the augmented algorithm provide mean link stress improvement on the order of 40%. An interesting general observation from the simulation experiments is that the length of the content providers' list (i.e. the number of alternative sources) is not of major importance neither for increased transfer throughput nor for reduced link stress. On the other hand, providing partial and diverse views of this list to different peers based on network-local knowledge coupled with provider selection based on minimum AS hop distance (ASP), consistently provides for faster content replication as well as for improved network resource utilization.

3 ALWAYS-ON PERFORMANCE MEASUREMENT AND CONTROL

An immensely important aspect of resilient networked systems is the provision of always-on predictable and self-managed services that facilitate the continuous operational optimisation of communication networks in an autonomic and dynamically adaptive manner. Such services can be enabled by employing the triptych of Measurement, Monitoring and Control (MMC), whose principal activities are concerned with assessing and assuring the infrastructural behaviour and the operational traffic dynamics in relatively short timescales. Quantitative measures of such temporal performance properties can be then used to provide the necessary input to control and adaptation algorithms, which ultimately facilitate a managed, autonomic, and optimised operation of the networked environment. Measurement and monitoring need to be always-on mechanisms to continuously report infrastructural and network components status, and most importantly, to assess the perceived performance of the operational traffic flows (at a local, network-wide, or even end-to-end level) a combinational and highly fluctuant attribute at potentially very short timescales. Then, feedback-based control algorithms can adapt to network traffic conditions and optimise certain metrics through operations such as adaptive routing.

In this chapter, we describe a *Diagnosis* and *Refinement* (DR) resilient system to form part of the outer resilience control loop, which employs *in-line measurement* to reveal the real-time performance of individual flows routed between two points in the network; it can then optimise routing based on minimising a per-flow cost metric.

We aim to bring our recent work on *in-line measurement* forward to this project, in order to demonstrate how the integration of control plane information flow and operations with the data path's main forwarding mechanism is not only beneficial, but also crucial for seamlessly incorporating resilience mechanisms in autonomic network structures and architecture. In-line measurement which is briefly described in the following subsections is a particular instantiation of the control/data plane integration that demonstrates how a real-time always-on instrumentation mechanism can be used to enable *autonomic* network performance diagnosis and refinement. In this regard, *autonomicity* mainly refers to the ability of the networked system (ANA compartment) to continuously monitor and measure its internal performance and subsequently, dynamically optimise the service delivered to the application flows based on adjusting certain control structures. The proposed system constitutes a compartment-wide *self-monitoring*, *self-organisation*, and *self-optimisation* resilience mechanism of the outer control (DR) loop that operates on an always-on manner without reacting on a specific external stimulus, such as an attack or a deliberate adverse operational condition, and minimises the need for operator intervention and static (re)configuration.

Figure 13 shows a high-level view of the always-on Diagnosis and Refinement measurement and control system operating over an ANA network compartment.

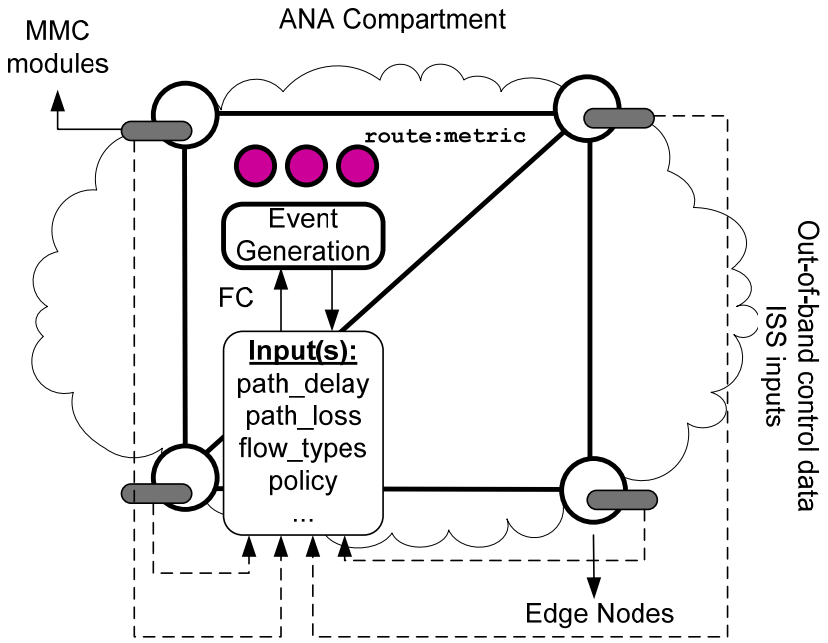

Figure 13: Compartment-wide DR system

Measurement and monitoring modules reside on the compartment's ingress and egress points, and instrument traffic in order to reveal the actual service experienced by the application flows routed over the network compartment at variable levels of aggregation. Measurement must be multi-point, in order to reveal how the traffic is routed within the compartment as opposed to how it originates or arrives at a single network point. In addition, instrumentation should be on the data-carrying traffic flows in order to measure the actual performance and integrate the measurement operations with the network's main forwarding mechanism, as it will be described in the next sub-section. We bring forward the in-line measurement paradigm for always-on traffic instrumentation that uses IPv6 protocol structures to facilitate ubiquitous measurement. We plan to engineer appropriate similar mechanisms to operate alongside specific ANA communication protocols.

From an operational perspective, real-time performance data collected at the edge nodes of the compartment will be processed by either centralised or distributed information flow components. A combination of options can be used to transmit this out-of-band control data, including push and pull models. Measurement information at defined levels of granularity will be treated as a particular type of Information Sensing and Sharing (ISS) inputs, which can be combined to form control events. For example, the different real-time performance characteristics experienced over the compartment's edge-to-edge paths, together with the flow types (and their consequent performance requirements) currently routed over the compartment, and potential administrative policies for traffic differentiation, can be different inputs to Functional Composition (FC) sub-systems, which will in turn generate optimisation events after resolving input priorities and potential conflicts [65].

In the remainder of this chapter we present a resilient system that enables adaptive routing based on temporal flow performance as this is reflected by a number of metrics depending on the sensitivities of particular applications (e.g. unidirectional delay vs. datagram loss). Routing protocol structures can then be updated using, for example, dynamic link-cost updates and per-flow routing policies. The system remains generic without specifying which individual routing protocol structures will be added/enhanced and its operation is evaluated on a simple multipath topology. We plan to integrate it in the future with routing structures developed specifically for the ANA architecture and possibly with certain existing (Internet) routing protocols.

3.1 Problem statement

Existing traffic measurement techniques and infrastructures fall into two main categories, namely active and passive techniques. Active measurement techniques inject additional traffic with known characteristics into the network to test particular attributes of a service [42][30][23][52], and they have been focusing on characterising properties of end-to-end network paths between instrumented systems. Passive measurements give highly accurate results by observing and analysing real traffic on a link without disruption to the service. They mainly operate at a single observation point within an administrative domain and try to provide feedback for network operations tasks [2][22][21][16].

Active and passive measurements rely either on the performance experienced by dedicated traffic or on the costly correlation of one-point observations to yield one-way performance results, and do not provide a framework for performing accurate and transparent service-quality measurements for different traffic flows that can be deployed on-demand in the network. We have developed the *in-line measurement* framework, a mechanism complementary to active and passive measurements, that tries to overcome the main limitations of both approaches by performing *targeted* measurement of the *actual* network traffic, deployed only where and when required.

At the same time, active and programmable networks research over the last decade has led to new developments in a number of areas ranging from secure programming languages [72][69], mobile code techniques [74], execution environments [74][27], active node platforms [54][44][33][34][64], service composition models [64][43][10], and so forth. Despite these valuable advances, the number of genuine applications where programmable network technologies are provably useful in real world networks is still limited. A large number of applications proposed so far [7][37][26] aim to demonstrate the functionality of certain active platforms, while others try to address problems that are best solved with conventional techniques such as (mobile) agents. Active networks are often regarded as a neat technology in seek of genuine applications, which would persuade operators that the benefit of network programmability exceeds the corresponding cost and risk involved in deploying and managing them.

We anticipate that active network services expose some properties that allow problems arising from e.g. network operation and management or service deployment, to be tackled in a more generic/elegant way. Being dynamically deployable on-demand, in a transparent and potentially automatic fashion at relevant points in the network, active services are suitable for a much wider range of applications/problems. At the same time,

the integration of a measurement mechanism that operates directly on the data path with active (dynamic, in this context) service deployment demonstrates operational automonicity of a network compartment with self-optimisation characteristics.

In this chapter we focus on the use of in-line traffic measurement [57] as an active service to facilitate dynamic routing link-cost updates that reflect fluctuations in traffic performance attributes [59]. Routing adjustment in response to varying service quality characteristics can improve overall network stability and performance, and presents a challenging task that really benefits from active and programmable service deployment.

In-line traffic measurements are used to assess the performance experienced by the flows along a transmission path, and measurement results are used to periodically adjust the network link costs in the routing protocol. As traffic measurements typically encompass a range of different characteristics (i.e. delay, jitter, packet loss, etc.), the calculation of link costs can be based on a multitude of different cost metrics. This allows for route optimisations tailored to specific applications or classes of applications with different QoS requirements (e.g. real-time synchronous vs. asynchronous applications).

Overall, this concrete system instantiates an always-on Diagnosis and Refinement (DR) mechanism, and demonstrates a network resilience mode of operation that does not react to an adverse operational condition per se, rather it continuously (self-)optimises network operation by conducting performance-based flow-level adaptive routing.

3.2 Multi-point in-line measurement to diagnose application performance

In-line measurement [57] is a technique to assess the QoS properties experienced by IPv6 flows accurately, independent of a particular network topology and transparent to the end-user applications. The direct measurement is carried out between two (or more) points in the network by piggybacking the relevant measurement indicators onto the actual data packets that are observed. IPv6 extension headers [18] allow Type-Length-Value (TLV)-encoded data to be inserted between the main IPv6 header and the upper (transport) layer header. Depending on which type of extension header is used for the traffic measurements (for example, destination options header or hop-by-hop options header [18]), one can control where and when to trigger the measurement activity. For example, in case the destination options header is used, traffic measurements will only be triggered end-to-end; whereas in the case of the hop-by-hop options header, any node along the transmission path could be involved. Moreover, the use of measurement information in the destination options header in conjunction with the routing header allows precise definition of where the traffic measurements in the network should take place.

The main benefit of this technique is that the traffic measurements are based on the actual user traffic rather than on general measurements based on other traffic flows. In addition to this property, by enforcing option processing only at identified nodes in the network and not hop-by-hop, in-line measurements eliminate the concern of instrumented packets being treated differently than the rest of the traffic in the network. Consequently, the measurements really reflect the performance experienced by the user data transmitted.

At the same time, the header extensions for the traffic measurements are defined by the network layer protocol itself, making the technique native, and equally applicable to any type of traffic (independent of the actual transport or user application).

Several measurement TLVs have been defined to be encoded within the IPv6 destination options header, which is examined by the final destination or optionally pre-defined intermediate nodes (based on the routing header) of a packet. Different TLVs implement a variety of performance metrics[1] by carrying packet departure/arrival timestamps, IP-based sequence numbers, trace information, etc. [58].

The clear separation of concerns between the measurement mechanism and particular analyses engines or post-processing measurement applications, constitutes in-line measurement a promising candidate-application for active and programmable networks; measurement instrumentation is deployed only where and when required, and the results are used as input for a variety of network operations tasks. Figure 14 shows different points along an end-to-end transmission path, where in-line traffic measurements can be deployed. End-systems as well as selected intermediate network nodes can be equipped with in-line measurement functionality. The node that starts the traffic measurement process inserts the desired extension header into the relevant data packets. These packets are then processed by the instrumented nodes along the transmission path. The measurement information is recorded, amended and/or extracted accordingly.

In this study, we focus on a particular application of in-line traffic measurements, whereby the measurements of up-to-date transmission characteristics (such as delay, jitter, packet loss and so forth) are used to dynamically update network link costs, so that routing decisions can be made more accurately.

Suitably selected active routers are used to deploy the measurement modules as they are needed, in order to adjust network link costs according to the current transmission performance. In the context of Figure 14, active routers can be nodes <B> and <C>, and the intermediate paths can be a point-to-point link or a virtual overlay link that spans across several hops in the underlying network.

The proposed traffic measurement application is particularly well-suited for dynamic and autonomic networks for a number of reasons:

- It is directly deployed on the data path
- It relies on direct access to data packets on the forwarding path (to support transparent measurements – independent from the applications)
- It is deployed, activated, and configured dynamically whenever and wherever there is need for it

[1] A set of performance metrics are defined within the IETF's IPPM WG

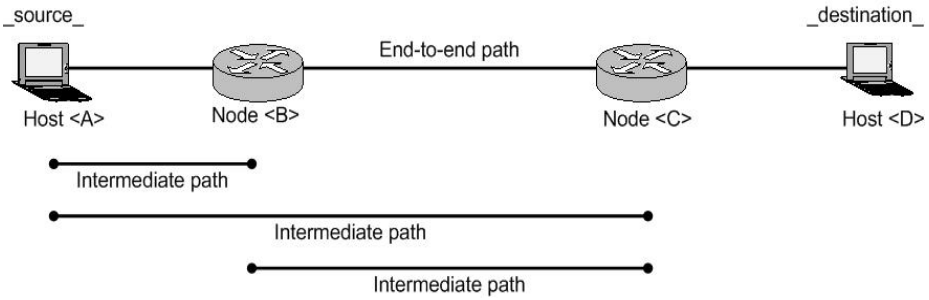

Figure 14: The different notions of end-to-end

These characteristics advocate the realisation of the add-on service as a dynamic service, offering the flexibility (and the ability) of on-demand deployment within the network.

Figure 15 provides a generic view of an ANA node that will host the measurement modules based on which link cost metrics will be updated through appropriate event generation and functional composition. Measurement modules will be installed and configured as MMC components whose granularity and modes of operation will be dynamically adjusted. Higher-level events can then be generated by composing certain measurement values and/or attributes. They can be a composition of values corresponding to different metrics and modules, or simply a periodic aggregate/average of a single measurement. Subsequently, an event can trigger a corresponding function to adjust a routing metric/policy that will ultimately alter the traffic flow to optimise application-perceived performance. This second level of indirection/composition serves the purposes of more complex decision-making based on multiple (and possibly diverse) inputs. For example, the update of a link cost metric may depend not only on temporal flow-level performance, but also on policies, service conditions, routing convergence, and infrastructural state. Hence a more complex classification graph that combines inputs from simultaneous and potentially conflicting events is deemed necessary for the autonomic node architecture. At the same time, the flexibility of being able to register multiple functional blocks (components) with the same set of events for potentially different purposes follows a modular system paradigm which is an asset for dynamic service composition and autonomic network operation.

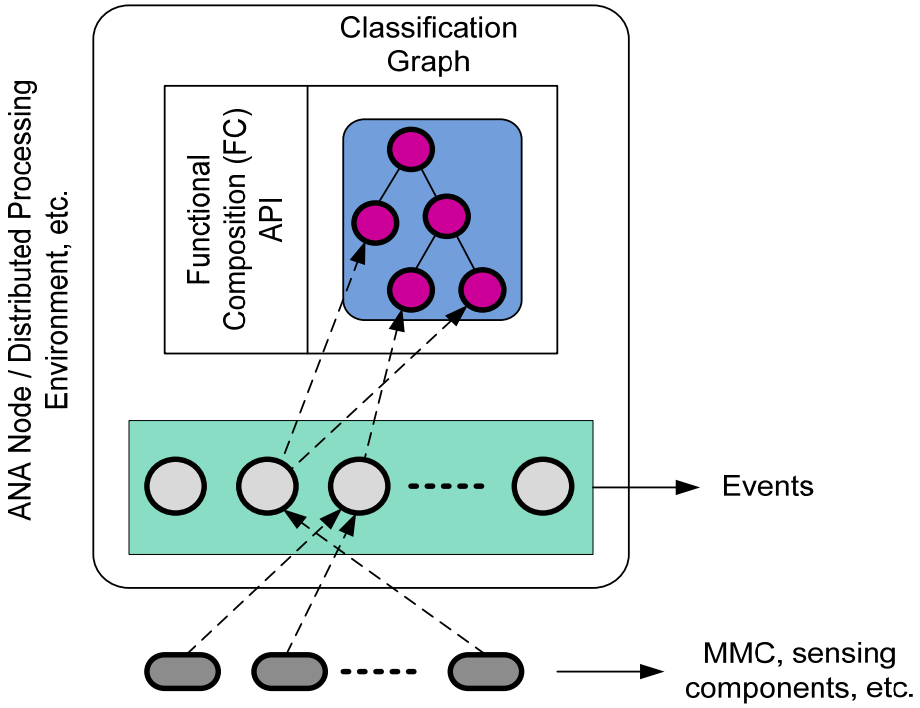

Figure 15: ANA Node; Generic Distributed Processing Environment

For the purposes of the current prototype, we have employed the LARA++ active node framework to host both the traffic measurement instrumentation and the link cost updates, due to its flexible and dynamic service composition properties. LARA++ [64] is a software implementation of a programmable router that is designed for commodity operating systems. It augments the functionality of a conventional router/host by exposing a programmable interface which allows dynamically programmable programs, referred to as active components, to provide network level services on any packet-based network. The architecture "hooks" directly into the router's operating system, and hence it enables the transparent interception of packets traversing the node. Intercepted packets can be processed by active components and then be re-injected back into the host OS for the default processing on the node. In this way, the functionality of a router's conventional network services can be flexibly extended (as opposed to altered), enabling lightweight augmentation of existing network services and allowing for gradual replacement of conventional router functionality. We consider this feature especially useful for our in-line traffic measurements, since it facilitates transparent processing of the relevant data traffic within selected nodes. The most relevant feature of LARA++ that constituted an ideal candidate for this prototype is its sophisticated model for service composition [63]. Each component that is to become part of the service composite on a running node installs one or several packet filters into nodes of a directed graph, referred to as the classification graph. These packet filters specify rules that the node uses to determine if traversing packets need to be processed locally. Once a packet is matched, it is delivered to the components that registered the filter for processing. The use of a configurable classification graph allows processing of packets of any type ranging from standard IP datagrams to completely bespoke packet formats. As a result, service

composition is defined implicitly by the classification graph and the packet filters installed by the active components. This type of composition approach provides a means to control both co-operation and competition among active components [63]. Figure 16 illustrates this concept in more detail. Packet filters are extremely flexible from the component developer point of view, because they enable the description of packets that are subject to active processing based on any bit or byte pattern. Yet, in most common cases it is sufficient to consider the flow information and/or the existence of specific header values in the packet. Packet filters are easily specified and installed by active components using an XML-based mark-up language.

The LARA++ classifier is conceptually very relevant to the design of the Autonomic Network Architecture (ANA) node, and indeed it is a candidate classifier for the ANA Functional Composition (FC) framework.

3.3 Refinement through adaptive routing

We have designed a modular system for Diagnosis and Refinement (DR) consisting of two main engines, one responsible for carrying the in-line measurement between two instrumented network nodes, and one for setting a maintaining a set of link cost metrics. The former has been implemented as a standalone active component that registers the relevant packet filters (depending on the flows of interest) at the IPv6 node of the classification graph [64].

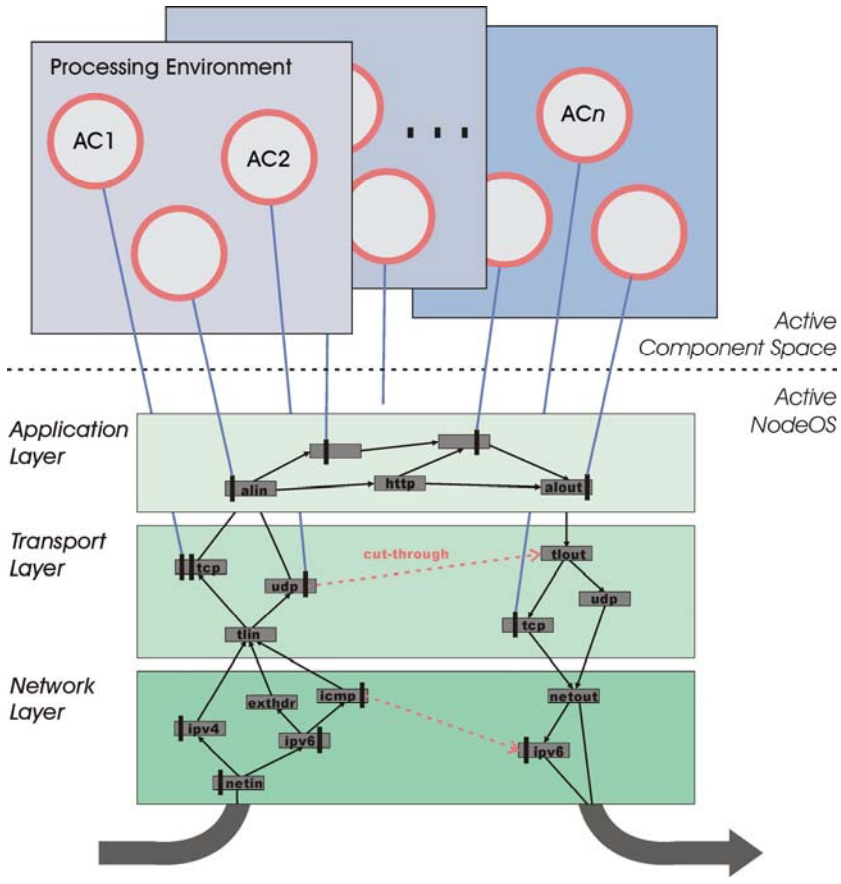

Figure 16: The LARA++ Classification Graph

It exposes an API that allows other components or user applications to use it. The latter module is a user-space application that reads and processes the measurement data, and maintains a data structure with separate metrics for the different types of measured attributes in the network (e.g. delay, jitter, packet loss, and so on). In order for the two modules to interface effectively, the evaluation module registers a callback interface with the measurements module. Periodically, the measurements module contacts the broker through the callback interface to stream the raw measurement data. Figure 17 illustrates the design of the proposed mechanism and shows how the two modules interface with each other.

The following sections describe in more detail the internal implementation and functionality of these two modules.

3.3.1 In-line measurements active components

The central functionality of this module is to perform the in-line measurements. As shown in Figure 17, this module consists of two main parts: i) a plug-in framework and ii) a set of measurement plug-ins. The plug-in framework provides the functionality for creating the appropriate destination options extension headers and for inserting/extracting the measurement data of previous nodes (which are encoded as TLV options).

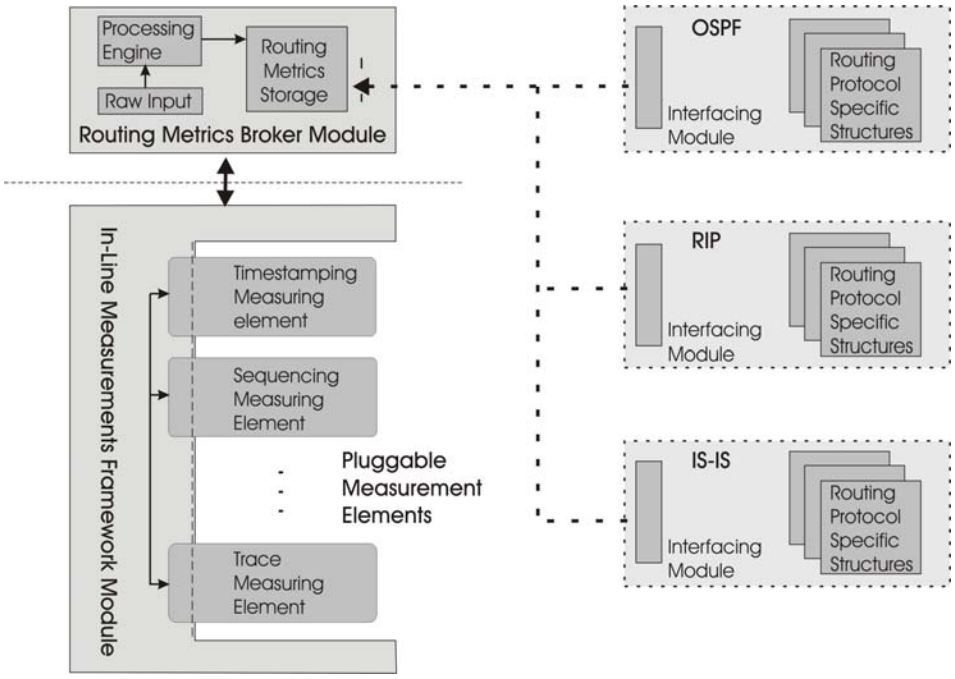

Figure 17: Basic Design of the Active Service

It exposes the necessary API for other applications to access the measurement data, to manage (add/remove/configure) the plug-ins, and to configure the filtering parameters. Filtering can be based on the source and destination addresses and ports, on transport protocol, traffic class, and on flow label values. The plug-ins are the code elements that carry out the actual measurements and generate the appropriate data that are inserted in the IPv6 packet header by the framework. Separate plug-ins are used for different types of measurements (e.g. transmission times and packet loss). The framework can accommodate several plug-ins simultaneously for performing different measurement (which results in more than one TLV options records in the IPv6 destination header), although this comes at the cost of reducing the data payload size. However, different plug-ins can create separate measurement TLVs for different data of interest.

The framework API provides an IOCTL-based interface for registering/attaching the plug-ins, and also adjusts the configuration parameters of the plug-ins such as the filtering and sampling granularity. The sampling rate can be configured by defining whether the module should instrument all packets matching the filtering criteria, 1-in-N, or act at a specific temporal sampling rate.

For the purposes of our prototype implementation, we have used two plug-ins to perform time and loss-related measurements accordingly. The first plug-in has been designed to measure one-way delay (OWD) between two points along a transmission path, as well as more synthetic time-related parameters such as jitter and throughput. This plug-in is used to insert and record departure and arrival timestamps of packets at the respective measurement nodes along the transmission path. The two measurement nodes synchronise their time through the Network Time Protocol (NTP) [45]. The second plug-in enables One-Way Loss (OWL) measurements by means of IP-based sequencing of packets. A source node inserts incremental sequence counters to packets belonging to the same flows, which are then observed at the destination. Packet loss as well as out of order delivery can be effectively measured by computing the differences of the TLV sequence numbers between successive packets. A flow in this context can be defined at different levels of granularity. At a fine granularity level, it can be the sequence of packets with the same source and destination IP addresses and transport ports. On the contrary, a flow can also be defined by all the packets traversing a certain point-to-point or virtual/overlay link. The next hop will also have to run the corresponding measurement module to keep track of sequence numbers as the packets arrive.

The in-line measurement active component registers the following packet filters with the classification graph: one or more filters for the outgoing packets of interest (the number of filters here depends on the filtering parameters configured by the measurement application) and one filter for the incoming IPv6 packets that contain our measurement header. Once a packet of interest is filtered, it is pulled out of the forwarding path and handed to our in-line measurement component. Depending on which filter captured the packet, TLV-encoded options are either inserted or extracted accordingly. The packets are then inserted back to the classification graph for further processing.

The information extracted from the incoming packets is delivered to the external broker module that has expressed interest in the respective measurement data. The role of the broker module is further described in the following section.

3.3.2 Routing metrics broker module

This control module accesses the in-line measurement active component in order to configure the in-line traffic measurements and collect the results. It is responsible for extracting and processing the appropriate raw measurement data, and for updating the costs table according to the routing metrics of interest. Node-local running routing protocols can then access these up-to-date cost metrics and optimise their routing information. In this way, routing protocols can always decide optimal routing paths based on up-to-date link quality information (with regard to the chosen metrics).

The current prototype of the broker module is implemented as a user-space application. The traffic measurement process starts by initialising the broker module where the user specifies the (virtual) link and the packet flows that should be used for the in-line traffic measurements. The user also selects which measurement plug-in instruments which flows. At start-up, the broker module instruments the installation and activation of the in-line measurement components on the respective active routers on both ends of the (virtual) link. Note that in the case of a virtual link (tunnel), the in-line measurements module will be installed several routing hops apart from each other, which enables the measurements for a whole routing path as opposed to a single physical link.

The broker module then establishes the necessary communication channels with the in-line measurement components, to pass configuration parameters such as the packet filters and sampling rate, and to receive the measurement results. Based on this data, the broker computes the appropriate link cost metrics that have been registered by the routing protocol(s) or other applications.

Once the in-line measurement component starts performing the measurements and delivering the measurement results, the broker module processes the data and updates the cost metrics data structure. This data structure stores single-value link costs for each measured attribute. It is accessible by the routing protocols through a “well-known” API so that they can update their internal data structures periodically, in order to reflect the dynamic link cost changes.

Since our main goal is to demonstrate the proposed functionality, our current implementation simply sets the cost values by averaging the N most recent measurements. More sophisticated calculations could be based on averaging a set of past cost values combined with the N most recent results or any other algorithm that would deliver a less fluctuating set of cost values.

Furthermore, since existing routing protocols typically do not use generic data structures among them nor do they share a common representation of link costs or routing metrics, those ones that want to benefit from the in-line traffic measurements have to be extended. As shown in Figure 17, we propose that the routing protocol will interface with our broker module through its own proprietary interface adapter. For example, the interface adapter for OSPFv3 would calculate OSPF-specific link costs from the measurement results and update the internal data structures accordingly.

The current prototype implementation is evaluated over a multipath topology containing a wireless and a multi-hop virtual link, and link costs are altered based on the temporal performance of the two candidate paths and the particular flows’ sensitivities to different metrics. Each link has a particular “cost” associated with performance metrics registered

by the set of active applications. A link can hence have a cost for packet loss, for unidirectional packet delay, for delay variation, etc. We haven't attempted to tailor our solution to a particular routing protocol implementation also due to the holistic view of link costs in current routing algorithm implementations that represent a single attribute associated with the link's physical characteristics or with a predefined policy. We rather aim at investigating how to enable measurement-based adaptive routing as a resilience mechanism to be integrated within the ANA node architecture and particular routing prototypes developed within the relevant tasks (T1.6, T2.1) of the project when these become more mature.

3.3.3 Evaluation

For the evaluation of our mechanism we used the IPv6 testbed infrastructure [47] at Lancaster University, where we have deployed two LARA++ active nodes (at points A and B) as illustrated in Figure 18. We created an artificial, yet realistic, network condition, where we stressed the ADSL uplink (at the tunnel connection) by generating 512-byte TCP/UDP traffic at an exponentially increasing rate of up to 62 packets per second. The wireless link on the other hand, being part of the campus WiFi network was subject to the usual (relatively congested) traffic encountered at midday hours.

We then triggered the installation of the proposed active service on node A and B according to the process described in section **Error! Reference source not found.**, and deployed the timestamping plug-in (for one-way delay and jitter) to instrument UDP traffic and the sequencing plug-in (for packet loss) for measurements on TCP traffic, respectively.

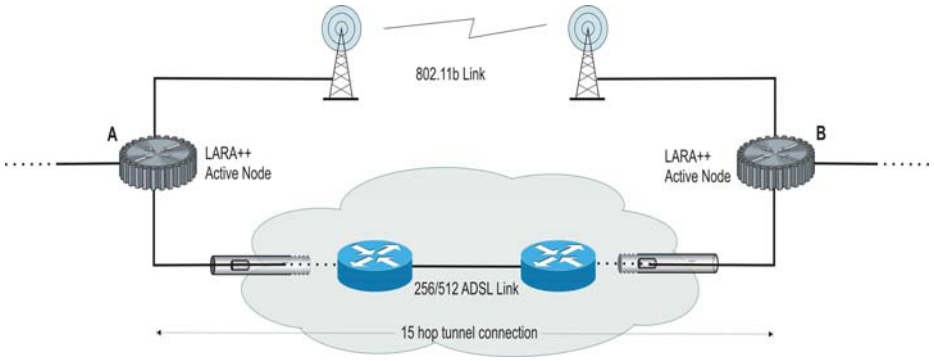

Figure 18: Experimental Network topologies

These choices are justified by the fact that TCP performance is known to be vulnerable to packet loss (continuous back-off), whereas UDP performance is impacted by increasing delays and delay variations (buffer adjustment requirements). After processing the measurements for appropriate time intervals using the broker module, we obtained the results illustrated in Figure 19 (ADSL downlink), Figure 20 (ADSL uplink) and Figure 21 (802.11b), for packet loss, delay and jitter. Table 1 summarises these results.

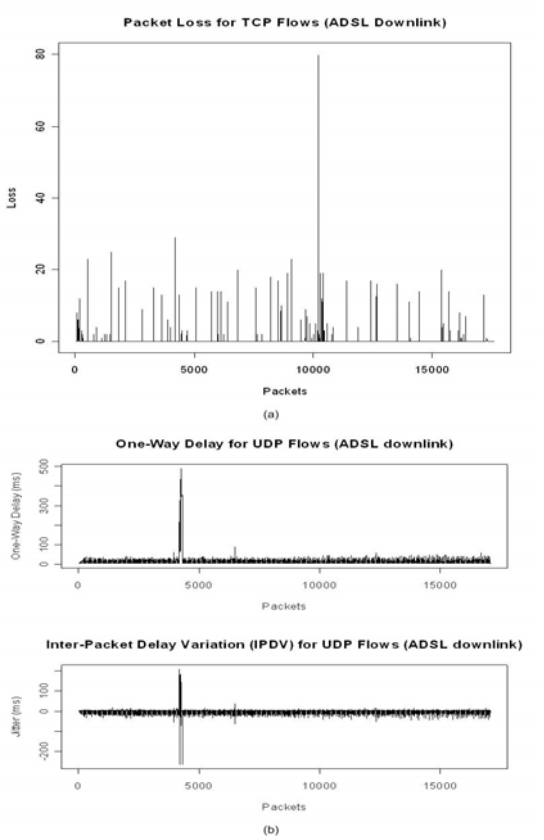

Figure 19: ADSL Downlink Measurements - (a) TCP Packet Loss (b) UDP OWD and Jitter

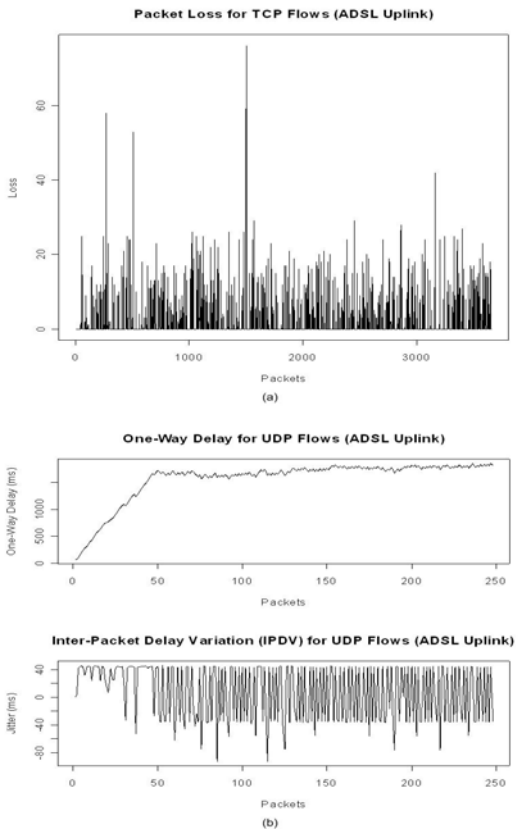

Figure 20: ADSL Uplink Measurements - (a) TCP Packet Loss (b) UDP OWD and Jitter

Comparing Figure 19 and Figure 21, we observed that the packet loss on the tunnel link is approximately 5% versus 4.8% for the WaveLAN link, yet the wireless link exposes more bursty characteristics. The delay experienced by the UDP flows over the tunnel connection was slightly better than on the WaveLAN link: the mean delay was 15.4 ms over the tunnel link versus 19 ms on the wireless link and 75% of the measurements yielded values less than 18 ms over the tunnel as opposed to 23 ms on the wireless link. Finally, jitter in both cases is almost the same, with most values laying between 1 and 2 ms. As a result, under the current congestion patterns, the WiFi network and the one (downlink) direction of the tunnel link exhibit similar characteristics, with the tunnel link having slightly better and more stable behaviour.

Figure 20 shows the performance of the ADSL uplink while becoming increasingly saturated by the artificially introduced data traffic. Under the very high stress, the tunnel link is hardly usable: 53.6% packet loss and rapidly increasing delays. This can be observed in the upper plot of Figure 20(b).

Under these traffic conditions, we can derive that the fittest routing configuration in our testbed is the asymmetric routing of traffic from A to B over the tunnel link and from B to A through the wireless link.

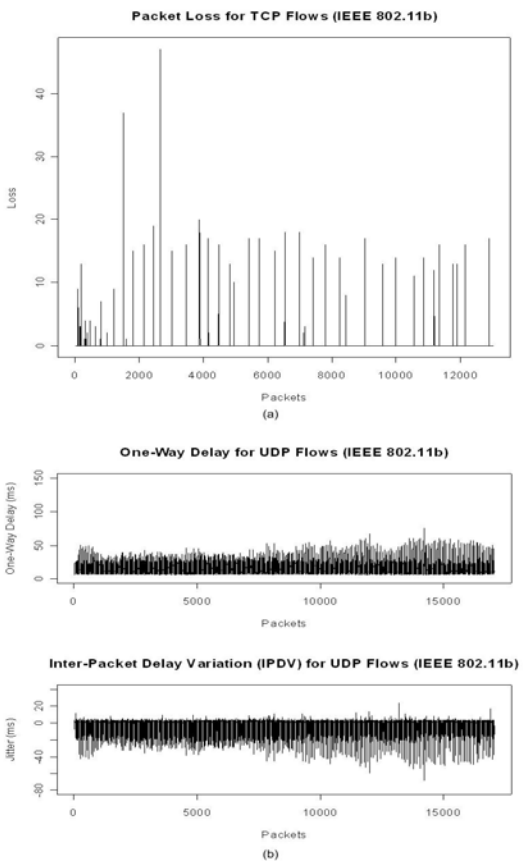

Figure 21: IEEE 802.11b Measurements - (a) TCP Packet Loss (b) UDP OWD and Jitter

Currently, according to the conventional operation of routing protocols, a router would select either the WiFi link or the tunnel connection to transport traffic between the points A and B. The choice would be based on static costs assigned to the two links based on their media type. In our setup it would always select (unless otherwise instructed) the WiFi link, since by default it is preferred over a virtual link, even if the latter had bigger capacity. This happens because the link cost assigned to the virtual link in absence of any other performance information is based on the distance metric. This default configuration can only change through the manual and static intervention of the administrator.

Based on our proposed mechanism, the routing protocol can dynamically adjust the link costs of both the tunnel and the wireless links based on the dynamic in-line traffic measurements (in response to their varying characteristics). In the case of our particular setup, the routing protocol would be able to detect the need for asymmetric routing and adapt the routing accordingly. Also, the ability of performing in-line measurements for specific classes of traffic, for example based on the transport protocol or Type-of-Service (ToS), is expected to benefit the implementation of ToS-based routing.

Table 1: Performance Statistics for the Different Links

	Delay		Jitter		Packet Loss
	Mean	75% Quantile	25% Quantile	75% Quantile	
ADSL Uplink	1558 ms	1768	-35 ms	44 ms	53.6%
ADSL Downlink	15.4 ms	18 ms	1 ms	2 ms	5 %
IEEE 802.11b	19 ms	23 ms	1 ms	2 ms	4.8 %

Existing work on QoS Routing research has considered adaptive routing based on dynamic cost metrics. Some early work focused on ToS routing [41], which is either

based on using multiple instances of routing protocols or maintaining routing tables with multiple metrics for different network attributes (i.e. delay, throughput, etc). Nevertheless, the link costs considered were static according to the natural characteristics of the link as it is the case with most routing protocols today. Other solutions that have been proposed in this area advocate the use of destination-driven/initiated routing path computations and updates towards them [14][19]. Clearly, these solutions are neither scalable nor pervasive, since they typically involve flooding mechanisms that cause both significant traffic overhead and high complexity. As a result, they suggest viable solutions only for maintaining routing paths to a small number of frequently used destinations. And, although they are quite dynamic, they often don't account for multiple metrics needed for the different attributes. A different approach for tackling congestion problems and therefore improving communication, has led to the idea of multipath routing. Probabilistic and other methods for load balancing traffic across multiple routing paths have been proposed [49] [9]. Although these solutions differ fundamentally from our approach, yet, we believe that our mechanism can complement these solutions to improve their performance through dynamic adaptation. Finally, the deployment of reconfigurable middleboxes or active network-based solutions has been considered in [40] and [71] in order to adapt or change the network configuration to match current traffic requirements. However, most of these solutions are not embeddable in general routing fabrics, but rather focus on out-of-band allocation of QoS resources in order to improve communication for individual flows.

4 CONCLUSION

In this document we have described the design and evaluation of diverse *self-contained* mechanisms whose operation provides and promotes the resilient operation of networked systems. Specific examples of modular systems were purposely selected to demonstrate the diverse ways in which resilience systems can operate over networked environments, and implement parts of the resilience strategy [68].

A real-time control-loop system has been designed to provide for **Detection**, **Defence**, **Remediation** and **Recovery** (D^2R^2) at the onset of adverse operational conditions, albeit arising from legitimate traffic patterns. And an always-on background **Diagnosis** and **Refinement** (DR) system was designed to provide for network traffic flow optimisation through adaptive routing, based on the continuous measurement of diverse service quality metrics over virtual network links.

Both these example systems demonstrate the two concentric modes of operation employed by diverse resilient mechanisms, namely the real-time reactive control, and the always-on management and optimisation. Future work will focus on investigating the integration of such resilient mechanisms and systems within the ANA node and network/compartment broader operation, whose prototypes is anticipated to have reached an advanced maturity level in the upcoming months.

5 REFERENCES

[1] Akamai, http://www.akamai.com/

[2] Apsidorf, J., Claffy, K., C., Thompson, K., Wilder, R., OC3MON: Flexible, Affordable, High Performance Statistics Collection, in Proceedings of the seventh Annual Conference of the Internet Society (INET'97), Kuala Lumpur, Malaysia, June 24-27 1997.

[3] Ari I., Hong B., Miller E., Brandt S., Long D., Modelling Analysis and Simulation of Flash Crowds on the Internet, Storage Systems Research Center, Jack Baskin School of Engineering, University of California, Santa Cruz, Stanta Cruz CA 95064, February 28, 2004

[4] Ari, I., Hong, B., Miller, E., L., Brandt, S., A., Long, D., D., E., Managing flash crowds on the Internet, IEEE/ACM International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'03), October 2003, Orlando, FL, USA

[5] Barabasi, A., L., Albert, R., Emergence of scaling in random networks, Science, pages 509–512, October 1999

[6] Barford, P., Plonka, D., Characteristics of network traffic flow anomalies, Internet Measurement Workshop (IMW'01), November 1-2, 2001, San Francisco, USA

[7] Bassi, A., Gelas, J-P., Lefevre, L., A sustainable Framework for Multimedia Data Streaming, in Proceedings of the 5th International Conference on Active Networks (IWAN), Kyoto, Japan, December 10-12, 2003.

[8] Bittorrent, http://bitconjurer.org/BitTorrent/

[9] Bohacek, S., Hespanha, J., P., Obraczka, K., Lee, J., Lim, C., Enhancing Security via Stochastic Routing, in Proceedings of the 11th International Conference on Computer Communications and Networks (ICCCN'02), Miami, Florida, October 14-16, 2002.

[10] Bossardt, M., Antik, R., H., Moser, A., Plattner, B., Chameleon: Realising Automatic Service Composition for Extensible Active Routers, in Proceedings of the 5th International Conference on Active Networks (IWAN), Kyoto, Japan, December 10-12, 2003.

[11] Boston university representative internet topology generator (BRITE), http://www.cs.bu.edu/brite/

[12] Bu, T., Towsley, D., On distinguishing between Internet power law topology generators, IEEE INFOCOM'02, New York, USA, June 23-27, 2002

[13] Characterization of Internet traffic loads, segregated by application, http://www.caida.org/analysis/workload/byapplication/

[14] Chen, J., Druschel, P., Subramanian, D., A New Approach to Routing with Dynamic Metrics, in Proceeding of IEEE INFOSOMM'98, San Francisco, USA, 29 March- 2 April, 1998.

[15] Chen, X., Heidemann, J., Flash crowd mitigation via adaptive admission control based on application-level observations, ACM Transactions on Internet Technology, Vol. 5, No. 3, August 2005, Pages 532–569

[16] Claffy, K., C., Miller, G., Thompson, K., The Nature Of The Beast: Recent Traffic Measurements From An Internet Backbone in Proceedings of the eighth Annual Conference of the Internet Society (INET'98), Geneva, Switzerland, July 21-24 1998.

[17] Cohen, B., Incentives build robustness in bittorrent. In Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA, June 2003

[18] Deering, S., Hinden, R., Internet Protocol Version 6 (IPv6) Specification, IETF, IPNG Working Group, RFC 2460, December 1998.

[19] Di Fatta, G., Gaglio, S., Lo Re, G., Ortolani, M., Adaptive Routing in Active Networks, Proceedings of IEEE OpenArch 2000, Tel Aviv, Israel, March 2000.

[20] Direct Connect, http://www.neo-modus.com/

[21] Feldmann, A., Greenberg, A., Lund, C., Reingold, N., Rexford, J., True, F., Deriving Traffic Demands For Operational IP Networks: Methodology And Experience, in Proceedings of ACM SIGCOMM'00, Stockholm, Sweden, August 28 – September 1 2000.

[22] Fraleigh, C., Diot, C., Lyles, B., Moon, S., Owezarski, P., Papagiannaki, D., Tobagi, F., Design and Deployment of a Passive Monitoring Infrastructure, in Proceedings of Passive and Active Measurement Workshop (PAM2001), Amsterdam, NL, April 23-24 2001.

[23] Georgatos, F., Gruber, F., Karrenberg, D., Santcroos, M., Susanj, A., Uijterwaal, H., Wilhelm, R., Providing Active Measurements as a Regular Service for ISP's, in Proceedings of Passive and Active Measurement Workshop (PAM2001), Amsterdam, NL, April 23-24 2001.

[24] Gnutella, http://www.gnutella.com/

[25] Gummadi, K., P., Dunn, R., J., Saroiu, S., Gribble, S., D., Levy, H., M., Zahorjan, J., Measurement, modelling, and analysis of a peer to peer filesharing workload, the biannual ACM Symposium on Operating Systems Principles (SOSP'03), October 19-22, 2003, NY, USA

[26] Hand, S., Harris, T., Kotsovinos, E., Pratt, I. Controlling the XenoServer Open Platform, in Proceedings of IEEE OpenArch'03, San Francisco, California, April 4-5, 2003.

[27] Hicks, M., W., Kaddar, P., Moore, J., T., Gunter, C., A., Nettles, S., PLAN: A Packet Language for Active Networks, In Proceedings of the 3rd ACM SIGPLAN International Conference on Functional Programming, pages 86-93, 1998.

[28] Izal, M, Urvoy-Keller, G., Biersack, E., W., Felber, P.A., Al Hamra, A., Garces-Erice, L.,. Dissecting BitTorrent: Five Months in a Torrent's Lifetime, Passive and Active Measurement Workshop (PAM'04), April 19-20, 2004, Antibes Juan-les-Pins, France

[29] Jung, J., Krishnamurthy, B., Rabinovich, M., Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites, the eleventh international World Wide Web conference (WWW'02), May 7-11, 2002, Honolulu, Hawaii, USA

[30] Kalidindi, S., Zekauskas, M., J., Surveyor: An Infrastructure for Internet Performance Measurements, in Proceedings of the ninth Annual Conference of the Internet Society (INET'99) INET'99, San Jose, California, June 22-25 1999.

[31] Karagiannis, T., Rodriguez, P., Papagiannaki, K., Should Internet service providers fear peer-assisted content distribution?, Internet Measurement Conference (IMC'05), October 19-21, 2005, Berkeley, CA, USA

[32] Kazaa media desktop, http://www.kazaa.com/

[33] Keller, R., Choi, S., Decasper, D., Dasen, M., Fankhauser, G., Plattner, B., An Active Router Architecture for Multicast Video Distribution. In Proc. of IEEE INFOCOM (3), pp 1137-1146, 2000.

[34] Keller, R., Ruf, L., Guindehi, A., Plattner, B., PromethOS: A Dynamically Extensible Router Architecture Supporting Explicit Routing, in Proceedings of the 4th International Conference on Active Networks (IWAN), Zurich, Switzerland, December 4-6, 2002.

[35] Keralapura, C., C., R., Taft, N., Iannaconne, G., Can ISPs take the heat from overlay networks? In ACM Workshop on Hot Topics in Networks (HotNets'04), November 15-16, 2004, San Diego, CA, USA

[36] Keralapura, R., Taft, N., Chuah, C. N., Iannaccone, G., Can. ISPs take the heat from Overlay Networks?, in Proceedings of. HotNets-III, November 2004

[37] Lefevre, L., Pierson, J-M., Guebli, S., Collaborative Web-Caching with Active Networks, in Proceedings of the 5th International Conference on Active Networks (IWAN), Kyoto, Japan, December 10-12, 2003.

[38] Leon-Garcia, A., Probability and random processes for electrical engineering, Addison-Wesley, second edition, 1994.

[39] Lua, E., K., Crowcroft, J., Pias, M., Sharma, R., Lim, S., A survey and comparison of peer-to-peer overlay network schemes, IEEE Communications Survey and Tutorial, November 2004

[40] Matta, I., Bestavros, A., QoS Controllers for the Internet, In Proceedings of the NSF Workshop on Information Technology, Cairo, Egypt, March 2000.

[41] Matta, I., Shankar, U., A., Type-of-Service Routing in Dynamic Datagram Networks, in Proceedings of IEEE INFOCOMM'04, Toronto, Ontario, Canada, June 12-16, 1994.

[42] Matthews, W., Cottrell, L., The PingER project: Active Internet Performance Monitoring for the HENP Community, IEEE Communications Magazine, Vol. 38, Issue 5, May 2000, pp. 130-136.

[43] Merugu, S., Bhattacharjee, S., Chae, Y., Sanders, M., Calvert, K., Zegura, E. Bowman and CANEs: Implementation of an Active Network, In Proc. of 37th Conference on Communication, Control and Computing, September 1999.

[44] Merugu, S., Bhattacharjee, S., Zegura, E., Calvert, K., Bowman: A Node OS for Active Networks, in Proceedings of IEEE INFOCOMM'00, Tel Aviv, Israel, March 26-30, 2000.

[45] Mills, D., Internet time synchronisation: the Network Time Protocol, IEEE Transaction on Communications, Volume 39, Issue 1, October 1991, pp. 1482-1493.

[46] MSNBC, http://www.msnbc.com

[47] MSRL – Mobile-IPv6 Systems research Lab". Research Project funded by Cisco Systems, Microsoft Research (Cambridge), and Orange Ltd, Lancaster University, 2001.

[48] Nakao, A., Peterson, L., Bavier, A., A routing underlay for overlay networks, ACM SIGCOMM'03, August 25-29, 2003, Karlsruhe, Germany

[49] Nelakuditi, S., Zhang, Z-L., Tsang, R., P., Du, D., H., C., Adaptive Proportional Routing and Localised QoS Routing Approach, in Proc. of IEEE INFOCOMM'00, Israel, March 26-30, 2000.

[50] Ng, T., S., E., Zhang, H., A Network Positioning System for the Internet, in USENIX'04, Boston, MA, June 27- July 2, 2004.

[51] Ng, T., S., E., Zhang, H., Predicting Internet networking distance with coordinates-based approaches. In Proceedings of IEEE INFOCOM, June 2002

[52] NLANR Active Measurement Project (AMP) Homepage, http://watt.nlanr.net//active/intro.html.

[53] Overnet/edonkey2000, http://www.edonkey2000.com/

[54] Paterson, L., Gottlieb, Y., Hibler, M., Tullmann, P., Lepreau, J., Schwab, S., Dandelkar, H., Purtell, A., Hartman, J., An OS Interface for Active Routers, IEEE Journal on Selected Areas in Communications, Volume 19, Issue 3, March 2001, pp. 473-487.

[55] Pezaros D. P., Mathy, L., Explicit Application-Network Cross-layer Optimization, 4th International Telecommunication Networking Workshop (IT-NEWS) on QoS in Multiservice IP Networks (QoS-IP 2008), Venice, Italy, February 13-15, 2008

[56] Pezaros, D., P., Cross-Layer Optimisation of Network Response at the Onset of Bursty Requests, in Proceedings of Multi-Service Networks (MSN'06), Cosener's House, Abingdon, UK, July 13-14, 2006.

[57] Pezaros, D., P., Hutchison, D., Garcia, F., J., Gardner, R., D., Sventek, J., S., In-line Service Measurements: An IPv6-based Framework for Traffic Evaluation and

Network Operations, in Proceedings of IEEE/IFIP NOMS 2004, Seoul, Korea, April 19-23, 2004.

[58] Pezaros, D.,P., Hutchison, D., Garcia, F.,J., Gardner, R., Sventek, J.,S., Service Quality Measurements for IPv6 Inter-networks, to appear in International Workshop on Quality of Service (IWQoS), Montreal, Canada, June 7-9, 2004.

[59] Pezaros, D., P., Sifalakis, M., Schmid, S., Hutchison, D., Dynamic Link Measurements using Active Components, in Proceedings of the Sixth International Working Conference on Active Networking (IWAN'04), Kansas, USA, October 27-29, 2004.

[60] Pouwelse, J., A., Garbacki, P., Epema, D.,H.,J., Sips, H.,J., A measurement study of the bittorrent peer-to-peer file-sharing system, Delft University of Technology Parallel and Distributed Systems Report Series, Technical Report PDS-2004-007, 2004.

[61] Pouwelse, J., A., Garbacki, P., Epema, D.,H.,J., Sips, H.,J., The bittorrent p2p file-sharing system: measurements and analysis, the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05), February 24-25, 2005, Ithaca, NY, USA

[62] Qiu, D., Srikant, R., Modelling and performance analysis of bittorrent-like peer-to-peer networks, ACM SIGCOMM'04, August 30-September 3, 2004, Oregon, USA

[63] Schmid, S., Chart, T., Sifalakis, M., Scott, A., C., "Flexible, Dynamic and Scalable Service Composition for Active Routers", In Proc. of IWAN 2002, pp 253-266, December 2002.

[64] Schmid, S., Finney, J., Scott, A., C., Shepherd, W., D., Component-based Active Network Architecture, IEEE Symposium on Computers and Communications, July 2001.

[65] Sifalakis, M., Louca, A., Hutchison, D., Peluso, L., Initial Design and Prototype Implementation of the Functional Composition Framework, "ANA Project Deliverable, D.2.4, January, 2008.

[66] Stading, T., Maniatis, P., Baker, M., Peer-to-peer caching schemes to address flash crowds, International Workshop on Peer-to-Peer Systems (IPTPS'02), March 2002, Cambridge, MA, USA

[67] Stavrou, A., Rubenstein, D., Sahu, S., A lightweight, robust p2p system to handle flash crowds, IEEE Journal on Selected Areas in Communications (JSAC), Vol. 22, Issue 1, January 2004, Pages 6-17

[68] Sterbenz, J., P., G., Schöller, M., Jabbar, A., Hutchison, D., First Draft of the resilience and Security Framework, Deliverable D3.2, Autonomic Network Architecture (ANA) Project, FP6-IST-27489/WP3/D3.2, February, 2007

[69] The Caml Language. Online Reference, INRIA, http://caml.inria.fr/.

[70] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/

[71] Vrontis, S., Sygkouna, I., Chantzara, M., Sykas, E., Enabling Distributed QoS Management Utilising Active Network Technology, in Proc. of Net-Con'03, October 2003.

[72] Wakeman, I., Jeffrey, A., Owen, T., Pepper, D., SafetyNet: A Language-Based Approach to Programmable Networks, in Computer Networks and ISDN Systems, 36 (1). 2001.

[73] Waxman, B., Routing of multipoint connections. IEEE J. Select. Areas Commun., December 1988

[74] Wetherall, D., J., Guttag, J., Tennenhouse, T., L., ANTS: A toolkit for building and dynamically deploying network protocols, in Proc. of IEEE Openarch, April 1998.