Towards Resilient Networks using Programmable Networking Technologies

http://www.comp.lancs.ac.uk/resilinets

Linlin Xie[1], Paul Smith[1],
Mark Banfield[3], Helmut Leopold[3],
James P.G. Sterbenz[2,1], and David Hutchison[1]

[1] Computing Department
InfoLab21
Lancaster University
Lancaster, LA1 4WA, UK
{linlin.xie,p.smith,jpgs,dh}@comp.lancs.ac.uk
[2] Information Technology and Telecommunications Research Center
Department of Electrical Engineering and Computer Science
University of Kansas
Lawrence, Kansas 66045-7621, USA
jpgs@ittc.ku.edu
[3] Telekom Austria AG
Lassallestraße 9
A-1020, Vienna, Austria
{mark.banfield, helmut.leopold}@telekom.at

Abstract. Resilience is arguably the most important property of a networked system, one of the three quality of service (QoS) characteristics along with security and performance. Now that computer networks are supporting many of the applications crucial to the success of the emerging Information Society – including business, health care, education, science, and government – it is particularly important to ensure that the underlying network infrastructure is resilient to events and attacks that will inevitably occur. Included in these challenges are *flash crowd* events, in which servers cannot cope with a very large onset of valid traffic, and *denial of service* attacks which aim to damage networked system with malicious traffic. In this paper, we outline the case for mechanisms to deal with such events and attacks, and we propose programmable networking techniques as the best way ahead, illustrated by a flash crowd example.

Key words: Resilience, Survivability, Disruption Tolerance, Programmable and Active Networking, Flash Crowd and Distributed Denial of Service (DDoS) Detection and Remediation, Quality of Service (QoS).

1 Introduction

Networks have become increasingly important in our daily lives, to the extent that we depend on them for much of what we do, and we are significantly disrupted when they cease to operate properly. Current networks in general, and the Internet in particular, do not provide the resilience that will be needed, especially when more critical applications depend on proper network operation.

Resilience is the ability of the network to provide and maintain an acceptable level of service in the face of various challenges to normal operation. These challenges include natural faults of network components (fault-tolerance); failures due to mis-configuration or operational errors; large-scale natural disasters (e.g., hurricanes, earthquakes, ice storms, tsunami, floods); attacks against the network hardware, software, or protocol infrastructure (from recreational crackers, industrial espionage, terrorism, or warfare); unpredictably long delay paths either due to length (e.g., satellite) or as a result of episodic connectivity; weak, asymmetric, and episodic connectivity of wireless channels; and high mobility of nodes and subnetworks. Addressing these challenges are required for network survivability [22]. We define resilience as survivability plus the ability to tolerate unusual but legitimate traffic load.

Note, that while attack detection is an important endeavour, it is in some sense futile, since a sufficiently sophisticated distributed denial of service (DDoS) attack is indistinguishable from legitimate traffic. Thus traffic anomaly detection that attempts to detect and resist DDoS attacks simply incrementally raises the bar over which attackers must pass. Since both cases adversely affect servers and cross traffic, as well as exhaust network resources, the goal is resilience regardless of whether or not an attack is occurring.

Resilient networks aim to provide acceptable service to applications, including the ability for users and applications to access information when needed (e.g., Web browsing and sensor monitoring), maintenance of end-to-end communication association (e.g., a video- or teleconference), and operation of distributed processing and networked storage. Resilient network services must remain accessible whenever possible, degrade gracefully when necessary, ensure correctness of operation (even if performance is degraded), and rapidly and automatically recover from degradation.

We believe that to realise resilient services it is necessary to have programmable networks – in particular, the ability of the network to dynamically adapt in response to learnt context information – providing the motivation for this need is the main contribution of this paper. In Section 2, we discuss in more detail the programmable networking features that are necessary for resilience and why they are necessary. We present in Section 3 an example resilient networking scenario – a flash crowd event, and show how programmable networking can be used to detect the onset of the ill-effects from such an event and how these effects can be mitigated. Recently, a number of important initiatives have emerged that aim to modify the Internet architecture, which could be used to realise resilient services; the rest of this section will present an overview of these initiatives.

1.1 Resilient Networking Initiatives

A knowledge plane (KP) [19] has been proposed to supplement the Internet architecture, which self-organises to discover and solves problems automatically. The principle is that a knowledge plane could reason based on collected information from all levels of the protocol stack to optimise applications, diagnose and tolerate faults and attacks, and make the network reconfigurable.

The KP would use cognitive AI to work on incomplete, inconsistent, or even misleading information, behave properly in the face of inconsistent high-level goals, and proactively work with new technologies and services. The KP can be considered a way of building resilient networks in the long-term future – the development of cognitive technology is still in its early stages and the KP highly depends on it. Furthermore, challenges need to be addressed in areas such as knowledge sharing (trust issues) and reasoning on vast amounts of information (scalability issues).

Work in the area of autonomic computing has largely focused on developing self-configuring, self-managing, and self-healing networked server systems [15]. There are now initiatives that consider making communications systems autonomic (e.g., [18, 17]). These communication systems aim to understand the context in which they operate, such as user requirements and network status, and then automatically adapt to meet service goals. Clearly, techniques for enabling autonomic communication systems are relevant for building resilient network services.

The COPS (Checking, Observing, and Protecting Services) project [20] aims to protect networks with devices called iBoxes, which perform observation and action operations at the network edge. COPS proposes to extend checking into the protocol domain, so that iBox functionality would migrate into future generations of routers. An *annotation layer* resides between the IP and transport layers for network management, which will allow annotated traffic to be appropriately processed.

2 Programmable Networks

Resilient networks need to be engineered with emergent behaviour to resist challenges to normal operation, recognise when challenges and attacks occur to isolate their effects, ensure resilience in the face of dependence of other infrastructure such as the power grid, rapidly and autonomically recover to normal operation, and refine future behaviour to better resist, recognise, and recover. We believe that programmable networking technologies will be a key enabler of the emergent and autonomic behaviour necessary for resilience.

The need for programmable networking technology [24–26] for building resilient networks stems from the nature of the challenges that will affect normal operation. These challenges will rapidly change over time and space. In other words, the moment in time when these challenges will threaten normal service operation will rapidly and arbitrarily differ, and over time new challenges will

emerge, such as new application traffic loads, forms of DDoS attacks, deployment environments, and networking technologies. Furthermore, the affected organisational entities and network services will change in an unpredictable manner. These characteristics preclude the use of a set of prescribed solutions to resiliency and mandate the use of a dynamically extensible infrastructure that can be aware of its environment.

The following subsections further catalogue and motivate the need for the programmable networking facilities that are required for resiliency.

2.1 Dynamic Extensibility and Self-Organisation

Programmability allows the network to respond to challenges by dynamically altering its behaviour and re-programming itself. This key ability of networks to change means that nodes do not need to be hard-coded or pre-provisioned with all the algorithms that may be needed to detect and respond to the challenges to normal operation. In fact, attempting to pre-program the complete set of resilience solutions is a futile exercise because of the dynamic and adaptive nature of the challenges to normal operation, as discussed earlier. Furthermore, we believe the network must be able to alter its behaviour without the intervention of network operators, because of the increasingly short timescales at which traffic patterns change (e.g., flash crowd) and attacks spread. Thus, it is essential that the network must be self-monitoring, self-diagnosing, self-reorganising, and self-managing.

In light of this, programmable networking devices must expose interfaces that allow their behaviour to be extended in a safe manner to appropriately privileged entities. Furthermore, a service that can be used to rapidly determine the most suitable programmable network locations to deploy resilience components must be available. For example, it should be possible for a resilient networking service to request the deployment of mitigation code in proximity to the source of a DDoS attack, even when the location of the source may be mobile. Approaches to this have been proposed in [28, 29], but much further work is required.

By introducing dynamic extensibility and self-organisation into the network, there is a risk of making the network unstable and potentially *worsening* the effect of any disruption to normal service provisioning. Furthermore, exposing interfaces that enable third-party services to understand and manipulate the operation of the network introduces a new entry-point for misuse. With this in mind, programmability and dynamic behaviour should be introduced carefully and exposed interfaces must be *stealthy* (i.e., not expose more functionality than strictly necessary). This is consistent with *moderate active networking* [16], in which the ability to inject and transport dynamic programming extensions is tightly controlled by the network service provider. Inter-providerAS relationships will have to be based on authentication and trust mechanisms.

2.2 Context Awareness

Understanding the characteristics of traffic and the topology in a resilient network is important. For example, when a DDoS attack occurs it is useful to learn the source addresses of the perpetrators so that remediation services can be invoked in appropriate network locations, including toward the source. In other words, it is important to understand the network context so that the correct remediation services can be invoked with the correct parameters. To understand network context it must be possible to inspect packets at line speed, as well as be aware of topology state and network signalling messages.

However, understanding network context is only one part of the picture. A resilient network should use context from a range of layers. Arguably, the *deeper* one can look into a packet at higher-layer protocol headers and data, the greater degree of information can be obtained, and more targeted any remediation service can be. Edge network devices are commercially available that are capable of application-level packet inspection at line speed (e.g., [13, 14]).

So that applications and services operating at different layers can understand one-another's context and work in harmony, interfaces that enable cross-layer interaction are necessary. Without understanding context across a range of layers, actions taken at one layer may not be complimentary at another. While it is clear about the motivation for cross-layer interaction, and there has been work in the context of specific parameters and protocols, there is no fundamental understanding of how this should be undertaken and what the benefits (performance and functional improvements) and costs (complexity and stability) are. A basic understanding of the nature of cross-layer interaction, resulting control loops, and its effect on the network needs to be gained [23].

3 Programmable Flash Crowd Detection and Mitigation

As an example to demonstrate how programmable networks can be used to build resilient services, in this section we describe an approach to detecting and mitigating the effects of a flash crowd event. To detect the ill-effects of a flash crowd event (e.g., a reduction in server response rate), we employ a mechanism that uses application and network-level information at a programmable edge device to detect a mismatch in anticipated and actual response rates from a server. We also discuss a number of approaches to mitigating the effects of flash crowd events by using the extensible nature of programmable networks.

3.1 Flash Crowd Detection

A flash crowd event [1], is characterised by a dramatic increase in requests for a service over a relatively short period of time, e.g., the sharp increase in requests for content on the CNN website immediately after the 9/11 attacks of 2001 [2]. These events can lead to a degradation or complete loss of service. It is important to detect the onset of a flash crowd event so that techniques to mitigate its effect can be invoked *before* a loss of service occurs.

A surge in service requests could cause a bottleneck to occur in the access network to the service provider, the systems providing the service, or both. In any case, one would expect to see a significant increase in request rate in a relatively short period of time and an associated levelling off or reduction in the response rate as the network queues or server resources become saturated with requests. This behaviour is what we aim to detect and use to trigger programmable mechanisms to protect the network.

The mechanism we propose detects flash crowd events that are targeted at Web servers. It makes use of application-level information, but performs the detection at the network level, and executes on a programmable edge router attached to the network that is providing the service. The mechanism inspects the volume of response traffic from a server, and based upon a difference between the expected volume of response traffic and the actual traffic, suggests the presence of a flash crowd event. In other words, if there is less response traffic than expected, we deduce the effects of a flash crowd event are beginning.

Proposals in [21] also compare estimated traffic volume to the actual volume to detect the onset of traffic volume anomalies. We use a similar idea, but do not aim to detect the presence of flash crowd events per se, but rather the onset of any ill-effects they cause. In [11], it is shown that Web traffic has self-similarity characteristics, in other words, the requested objects follow a power-law distribution. We use this fact and the content-size distribution of requested objects, learnt from sampling the content-length field in HTTP response headers, to estimate the volume of response traffic.

Normally, the sum of the sizes of the requested objects would form the response traffic volume, as shown in Equation 1, where v is volume of response traffic, r is the number of requests, and S_i is the size of the object associated with a request r_i. We maintain the average incoming HTTP request rate for a server and use this along with the learnt content-size distribution to estimate the volume of response traffic expected. Equation 2 shows how we calculate the Exponentially Weighted Moving Average (EWMA) incoming request rate (f), where c is the request rate at a given point in time. Equation 3 describes how we use the integer value of this average (f) to calculate the expected volume of response traffic (e) at time t, where G_i is the estimated content-size for a request f_i. By selecting an appropriate values for α, we aim to obtain a close estimate of the response traffic volume.

$$v = \sum_{i=1}^{r} S_i \tag{1}$$

$$f(t) = (1 - \alpha) \times f(t-1) + \alpha \times c(t), \quad with \quad \alpha > 0 \tag{2}$$

$$e(t) = \sum_{i=1}^{\lfloor f(t) \rfloor} G_i \tag{3}$$

$$\{\frac{a(t)}{e(t)} | \, t = 1 \ldots n\} \sim N(\mu, \sigma^2) \tag{4}$$

The ratio of the observed response traffic volume (a) to the estimated traffic volume (e) should follow a normal distribution: $N(\mu, \sigma^2)$, see Equation 4. The value of μ should be slightly greater than one, because we did not include the TCP/IP header size in our calculations. We use the EWMA of the ratio to smooth fluctuations caused by inaccuracies in the guessing mechanism. In Section 3.2, we show how we test the ratio distribution and calculate the parameter for the distribution and gain confidence (95% in this case), from which we can ascertain if the effects of a flash crowd are occurring. If continuous points are observed to be beyond the confidence range, it suggests the occurrence of abnormality.

3.2 Simulation of Flash Crowd Mechanism

To give an indication of the effectiveness of the flash crowd detection mechanism, we simulated such an event using ns-2. HTTP traffic was generated using the PagePool/WebTraf application in ns-2. Parameters used for generating HTTP sessions follow the distributions presented in [9]. The request rate for background traffic was modified to be approximately 150 requests/sec and flash traffic to 1200 requests/sec. The request rate of flash traffic was set to be almost eight times greater than that of background traffic, which is modest for a flash crowd event as the hit-rate for CNN just after 9/11 was twenty times its normal rate [2]. The parameters used for the background and flash traffic are shown in Table 1.

Table 1. Simulation parameters

Traffic Type	Number of Sessions	Inter-session Time [s]	Number of Pages per Session	Inter-page Time [s]	Number of Objects per Page	Inter-object Time [s]	Object Size [KB]
Background	1000	1	15	1	10	0.01	Avg:12 Shape: 1.2
Flash Crowd	20000	0.025	10	1	10	0.01	Avg:12 Shape: 1.2

The simulations ran for 1200 seconds; flash traffic started at 500 seconds. We used a simple network topology, this included twenty clients, an ingress edge router, an egress edge router, and a server. The bandwidth of the links between the clients and the ingress router were set to 10 Mb/sec, the two routers were connected by a 50 Mb/sec link, and the egress router was connected to the server by a 15 Mb/sec link. A detection interval (how often we checked the ratio of actual (t) to expected (e) response traffic volumes) was set to 30 seconds and the α value was set to 0.2. The simulation with the same configuration was run three times and the mean values were used for generating the graphs.

The onset of the flash crowd event can be seen at 500 seconds into the simulation in Figure 1. At 1020 seconds, the request rate starts to drop due to the sessions running out. Figure 2 shows how the normal response rate is around 1.6 Mb/sec and during the flash crowd event it reaches and stabilises at 1.8 Mb/sec. The stabilisation of the response rate is caused by the buffers on the server's ingress link becoming saturated and subsequently dropping incoming requests.

Fig. 1. Request rate during a flash crowd event starting at 500 seconds

Fig. 2. Response traffic rate during a flash crowd event starting at 500 seconds

To gain an estimate of μ and σ for the normal distribution of the ratio, we ran ten thousand background traffic sessions using the parameters shown in

Fig. 3. The distribution of sampled ratios in normal situation

Table 1. The value of μ was set to the average of the samples: 1.10817, and σ to the standard deviation of the samples: 0.227477. Figure 3 shows that the sample distribution appears close to the normal distribution of $N(\mu, \sigma^2)$. The 95% confidence range of this distribution is [0.662315, 1.554025], which means that the possibility of the ratio value going beyond this range is 5%. We use more than two continuous values outside the confidence range to detect the saturation of the server side.

Figure 4 shows that the ratio drops shortly after the onset of the flash crowd event and subsequently oscillates around 0.2 with a small amplitude. Recall that two continuous ratio values outside the confidence range [0.662315, 1.554025] is used to diagnose that the effects of a flash crowd are being felt. Given this, with a detection interval of 30 seconds, saturation can be confirmed at 570 seconds.

Fig. 4. Ratio of actual response traffic amount over estimated traffic

3.3 Flash Crowd Mitigation Mechanism

To protect a Web server and the cross traffic in the network, we propose two strategies. The first is to drop requests that the server side is not able to manage at the ingress points of a provider's network. The ingress points are discovered by routers that perform a *pushback* mechanism, the basic concept and mechanism of which are presented in [7][8]. In summary, with a slight variation, our mechanism is invoked on the server's edge router which identifies the incoming interfaces of aggregates of high volume of requests to the server. The router then sends messages to the immediate upstream routers (from which the high aggregate request volumes came from) to recursively carry out this procedure and push back requests until the provider's ingress router is reached. The second strategy is to re-route response traffic inside the network to improve the traffic distribution and reduce the possibility of links becoming congested. The reason for the first action is straightforward – to push request traffic that cannot be served outside of the network to save network resources. The reason for the other action and the mechanisms to do it are described below.

According to [12], an important metric for measuring how well traffic is distributed in a network is *maximum utilisation*. Larger maximum utilisation values indicate that links are more sensitive to bursts. Large amounts of flash crowd traffic would cause heavily skewed distribution in the network, which could reduce the quality of service for cross traffic. To have a better distribution we need to reduce the maximum utilisation. These strategies are the subject of future work, as discussed in Section 4.

3.4 Related Flash Crowd Detection and Mitigation Work

The implications of flash crowd events and DoS attacks for Web sites and content distribution networks (CDNs) are discussed in [3]. They propose enhancements to CDNs that make them more adaptive and subsequently better at mitigating the effects of flash crowd events.

Collaborative caching mechanisms that can be used to redirect requests to appropriate caches in light of a flash crowd event are proposed in [4]. The challenge here is to make sure that the appropriate content is cached – this may be difficult to predict.

The authors of [5] describe a mechanism that breaks content up into small pieces and returns each request a piece. Clients need to talk to each other for other pieces of the content. This mechanism requires servers to perform the content manipulation, and requires modifications to Web browser applications and HTTP protocol.

In [6] the problems associated with flash crowds are addressed by making changes to the architecture of Web servers – approaches that allow dynamic resource allocation across multiple servers are proposed. We address the flash crowd problem from the point of a network service provider (and potentially also a third-party application service provider), and make no assumption about the Web server architecture in use.

An approach to dropping requests at the server's ingress point to a network is proposed in [9]. The rate at which requests are dropped is set dynamically. A major drawback of this approach is that it requires the inspection of application layer headers of each packet. We have shown here that you can do detection at the network level while only sampling the application layer headers occasionally.

4 Future Work

Because our flash crowd detection mechanism uses *hints* to determine the onset of the ill-effects of a flash crowd event – it guesses the expected volume of response traffic – there is a possibility it could give false positives. Further investigation is necessary to determine under what conditions this could occur and what effects a false positive may have. In our simulations, we set the detection interval to 30 seconds; investigating whether we could effectively reduce this interval to enable faster detection is something we plan to investigate through further simulation.

As part of future work into mitigating the effects of flash crowd events, we propose to improve the distribution of response traffic by instigating multi-path routing for traffic that is tolerant to packet mis-ordering. A way to approach this is by a server's edge router building a multi-route database, in which all possible routes between the server's edge router to all the other edge routers along with the available bandwidths are held. The database is built by deploying active code to collect routing information and available bandwidth information from programmable routers. When the server's edge router observes or is informed that the response traffic is consuming too much bandwidth on one of the links, it could distribute the traffic over a number of routes. An approach such as this removes the need to change existing routing protocols, as in [12], which manipulates the link weights in the OSPF routing database.

Investigating aspects of resilience in the context of computer networks is an emerging research topic. In the recently funded Autonomic Networking Architecture (ANA) EU research project [18], we will investigate the the use of resilience techniques and mechanisms to support autonomic networks.

5 Conclusions

In this paper, we have presented work in progress in the important area of the resilience of networked systems. In addition to presenting the basic argument that resilience is really needed in the modern networked world, we argue for programmable networking techniques as an appropriate way ahead to build resilience mechanisms.

By means of a modest flash crowd example, we outline simulation results that aim to show the promise of programmable networking in this crucial area. Furthermore, the mechanism demonstrates that multi-layer cooperation is a useful tool to enable resilient networks. The simulation results indicate that our detection mechanism for flash crowd events has potential.

Future work will focus on the mitigation of flash crowd events and also DDoS detection and repair. By focusing on a particular application scenario we aim to develop and prove a resilient network architecture that uses programmable networking technologies.

Acknowledgements

Linlin Xie and Paul Smith are supported by Telekom Austria. We are grateful to Steven Simpson for his help and contributions with the simulations. We also appreciate the comments from the anonymous reviewers.

References

1. Larry Niven, "Flash Crowd", in *Flight of the Horse*, Ballantine Books, September 1973
2. William LeFebvre, "CNN.com: Facing A World Crisis", http://www.tcsa.org/lisa2001/cnn.txt, 2001
3. Jaeyeon Jung, Balachander Krishnamurthy, Michael Rabinovich, "Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites", Proceedings of The Eleventh International ACM World Wide Web Conference (ACM WWW 2002), Hawaii, USA. May 2002
4. Tyron Stading, Petros Maniatis, Mary Baker, "Peer-to-Peer Caching Schemes to Address Flash Crowds", Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS 2002), Cambridge, USA. March 2002
5. Jay A. Patel, Indranil Gupta, "Overhaul: Extending HTTP to Combat Flash Crowds", Proceedings of the 9th International Workshop on Web Caching and Content Distribution (WCW 2004), Beijing, China, October 2004
6. Abhishek Chandra, Prashant Shenoy, "Effectiveness of Dynamic Resource Allocation for Handling Internet Flash Crowds", University of Massachusetts Technical Report, TR03-37, 2003
7. Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, Scott Shenker, "Controlling High Bandwidth Aggregates in the Network", ACM SIGCOMM Computer Communication Review, Vol.32, No.3, pp.62-73, July 2002
8. John Ioannidis, Steven M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks", AT&T Technical Report, December 2001
9. Xuan Chen, John Heidemann, "Flash Crowd Mitigation via Adaptive Admission Control Based on Application-Level Observation", USC/ISI Technical Report, ISI-TR-2002_557 (revised version), March 2003
10. Jelena Mirkovic, Peter Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms", ACM SIGCOMM Computer Communications Review, Vol.34, No.2, pp.39-53, April 2004
11. Mark E. Crovella, Azer Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes", In IEEE/ACM Transactions on Networking, Vol.5, No.6, pp.835-846, December 1997
12. Bernard Fortz, Mikkel Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights", Proceedings of the 19th Conference on Computer Communications (INFOCOM 2000), Tel-Aviv, Israel, March 2000

13. Bivio Networks, http://www.bivio.net/
14. IBM BladeCenter, http://www-03.ibm.com/servers/eserver/bladecenter/
15. IBM Autonomic Computing, "White Paper: An architectural blueprint for autonomic computing", Third Edition, http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf, June 2005
16. Alden W. Jackson, James P.G. Sterbenz, Matthew N. Condell, Regina Rosale Hain, "Active Network Monitoring and Control: The SENCOMM Architecture and Implementation", 2002 DARPA Active Networks Conference and Exposition (DANCE'02), pp.379, 2002
17. The Autonomic Communications Forum, http://www.autonomic-communication-forum.org/
18. The Autonomic Networking Architecture (ANA) research consortium, http://www.ana-project.org/
19. David Clark, Craig Partridge, J.Ramming, John Wroclawksi, "A Knowledge Plane for the Internet", Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 2003), Karlsruhe, Germany, August 2003
20. Randy Katz, George Porter, Scott Shenker, Ion Stoica, Mel Tsai, "COPS: Quality of Service versus Any Service at All", Proceedings of the Thirteenth International Workshop on Quality of Service (IWQoS 2005), Passau, Germany, June 2005
21. Anukool Lakhina, Mark Crovella, Christophe Diot, "Diagnosing Network-wide Traffic anomalies", Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM 2004), Portland. Oregon, USA, August 2004
22. James P.G. Sterbenz, Rajesh Krishnan, Regina Rosales Hain, Alden W. Jackson, David Levin, Ram Ramanathan, John Zao, "Survivable Mobile Wireless Networks: Issues, Challenges, and Research Directions", Proceedings of the ACM Wireless Security Workshop (WiSE) 2002 at MobiCom, Atlanta GA, September 2002, pp. 31-40
23. James P.G. Sterbenz and David Hutchison, "Towards a Framework for Cross-Layer Optimisation in Support of Survivable and Resilient Autonomic Networking", Dagstuhl Seminar 06011, January 2006.
24. Ken Calvert, Samrat Bhatacharjee, Ellen Zegura, James P.G. Sterbenz, "Directions in Active Networks", IEEE Communications, Vol.36 No.10, pp.72-78, October 1998
25. David L. Tennenhouse, David J. Wetherall, "Towards an Active Network Architecture", ACM Computer Communication Review, Vol.26, No.2, pp.5-17, April 1996
26. David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, Gary J. Minden, "A Survey of Active Network Research", IEEE Communications Magazine, Vol.35, No.1, pp.80-86, January 1997
27. Stefan Schmid, "A Component-based Active Router Architecture", PhD Thesis, Lancaster University, November 2002
28. Paul Smith, "Programmable Service Deployment with Peer-to-Peer Networks", PhD Thesis, Lancaster University, September 2003
29. David Spence, Jon Crowcroft, Steven Hand, Tim Harris, "Location Based Placement of Whole Distributed Systems", Proceedings of ACM Conference on Emerging Network Experiment and Technology (CoNEXT 2005), Toulouse, France, October 2005, pp.124-134