A Cross-Layer Architecture For Autonomic Communications

M.A. Razzaque[1], Simon Dobson and Paddy Nixon

Systems Research Group
School of Computer Science and Informatics
UCD Dublin IE
abdur.razzaque@ucd.ie

Abstract. Layered architectures are not sufficiently flexible to cope with the dynamics of wireless-dominated next-generation communications. Most existing architectures and approaches depend purely on local information and provide only poor and inaccurate information gathering at the global scale. De-layered or cross-layer architectures may provide a better solution: cross-layering allows interactions between two or more non-adjacent layers in the protocol stack. We propose a new cross-layer architecture which provides a hybrid local and global view, using gossiping to maintain consistency. We evaluate our proposal informally in terms of communication complexity and in terms of its ability to support the "self-*" properties being proposed within the autonomic communications community.

Keywords: Cross-Layer Architecture, Autonomic Communications, Gossiping.

1 Introduction

The worldwide success of the Internet is dominated by the layered architecture, but a strict layered design is insufficiently flexible to cope with the dynamics of wireless-dominated next-generation communications. Recent studies [1] show that careful exploitation of some cross-layer protocol interactions can lead to more efficient performance of the transmission stack – and hence to better application layer performances – in different wireless networking scenarios.

Cross-layer design breaks away from traditional network design, where each layer of the protocol stack operates independently and exchanges information with adjacent layers only through a narrow interface. In the cross-layer approach information is exchanged

[1] This work is partially supported by Science Foundation Ireland under grant number 04/RP1/1544, "Secure and Predictable Pervasive Computing."

between non-adjacent layers of the protocol stack, and end-to-end performance is optimized by adapting each layer against this information Cross-layering is not the simple replacement of a layered architecture, nor is it the simple combination of layered functionality: instead it breaks the boundaries between information abstractions to improve end-to-end transportation.

One obvious shortcoming of the strict layering is the lack of information sharing between protocol layers. This hampers optimal performance of the networks, since shared layer information is the prerequisite for many forms of performance optimization. On the other hand, cross-layer systems shift the research landscape away from optimizing the performance of individual layers, and instead treat optimization as a problem for the entire stack. The technique consists of taking into account information available from different levels [2], not necessarily adjacent, in order to create a system much more sensitive to its environment, load and usage. Essentially we replace the narrow inter-layer interface with a wider, richer and more holistic view of the network's issues, goals and constraints.

The assumptions in the wired IP stack are inadequate for wireless networking, and TCP/IP is known to suffer from performance degradation in mobile wireless environments. This is because such environments are prone to packet losses due both to high bit error rates and mobility-induced disconnections. TCP interprets all packet losses as an indication of congestion and (inappropriately) invokes congestion control mechanisms, which leads to degraded performance. Cross-layering between Link Layer and Transport Layer is an easy solution to this problem [3].

On the other hand, it is clear from the recent initiatives in autonomic computing and autonomic communications [4, 5] that there is a need to make future networks self-behaving, in the sense that they work in an optimal way with "endogenous" management and control, with minimum human perception and intervention. Autonomic communication is the vision of next-generation networking which will be a self-behaving system with properties such as self-healing, self-configuration, self-organization, self-optimization and so forth – the so-called "self-*" properties. To attain this self-behaving system within existing strictly-layered approaches may be possible to certain extent, but will not (we claim) leverage all the possible optimizations. We consider cross-layer architectures to be better suited to achieving the self-optimization, self-configuration and other "self-*" properties targeted by autonomic approaches.

Most existing architectures (including GRACE [6], WIDENS [7], MobileMan [8]) rely on local information and views, without considering the global networking context or views which may be very useful for wireless networks in optimizing load balancing, routing, energy management, and even some self-behaving properties like self-organization. Only CrossTalk [9] is based on a (even partially) global view of the network. On the other hand, POEM [10] is the only architecture considering self-optimization that could be helpful for autonomic communication. Collecting and maintaining network-wide, global statistics can be expensive, while global actions are hard to co-ordinate; however, the effects of such systems can often be dramatic, and they can address problems that are difficult to detect, diagnose or solve using purely local information. To explore the impact of this idea, we propose a new cross-layer architecture

based on building and maintaining a global view of the network's state and constraints, utilizing gossiping for gathering information from neighboring nodes. For scalability and overhead issues, we limit gossiping processes among direct neighbors.

Section 2 describes related study on cross- layer architectures. Our contribution, a cross-layer architecture for autonomic communications is presented in section 3. Section 4 briefly describes the difficulties implicit in cross-layer interactions. And section 5 concludes with some directions for future work.

2 Related Study

Research on cross-layer networking is still at a very early stage, and no consensus exists on a generic cross layer infrastructure or architecture. However, the importance of a sound architecture to handle the proliferation of cross-layer operations in wireless as well other communications media is clear, especially in autonomic systems for which properties need to be specified and maintained with minimal manual configuration and intervention [11]. A number of proposals for cross layer designs and their corresponding architectures have been published in the literature. Most of these architectures are relying on node-local view and very few utilize both the local view as well as network-wide global view.

GRACE (Global Resource Adaptation through Co-opEration) [6] is a cross layer adaptation framework. All system components (hardware, network, and operating system) and applications are allowed to be adaptive. These adaptive entities co-operate with each other to achieve a system-wide optimal configuration, for example to maximize system utility, in the presence of changes in the available resources or application demands. However, its cross-layer approach includes no explicit consideration of cross layering within the networking layers or protocol stack. WIDENS (Wireless DEployable Network System) [7] has been proposed with an aim to acquire the interoperability, cross layering and re-configurability at the same time. This cross layering architecture seems a promising one where protocol optimization is based on the local state information but it is still in the validation stage and so lacks any real measurement of efficiency especially in terms of performance.

The MobileMan [8] architecture presents, along with the strict layering, a core component, *Network Status*, which functions as a repository for information that uniformly manages the cross-layer interaction while respecting the principle of dividing functionalities and responsibilities in layers. The approach aims to optimize overall network performance with respect to local state information by increasing local interaction among protocols, decreasing remote communications, and consequently saving network bandwidth. Performance improvement verifications are yet to be published. ECLAIR [12] is local-view-based, efficient cross-layer architecture for wireless protocol stacks. Along with legacy protocol stack it consists of two main components: an *Optimizing Sub-System*, the cross layer engine which contains many *Protocol Optimizers*, which are the "intelligent" components of it, and *Tuning Layers* provide the necessary

APIs to the protocol optimizers for interacting with various layers and manipulating the protocol data structures. There is no processing overhead on the existing stack since the optimizing subsystem executes in parallel to the protocol stack.

CrossTalk [9] is the only (as far as we are aware) cross-layer architecture, which has the ability to reliably establish a network-wide, global view of the network under multiple metrics. Having such a global view, a node can use global information for local decision processes in conjunction with a local view containing node-specific information contributed by each layer of the stack or system component. To keep overheads low, no additional messages are sent: instead the local information taken from the local view is piggybacked onto outgoing packets. Piggybacking implies that it is quite unlikely that any node will obtain fully accurate global view under many likely models of data exchange. Even with an uncertain and poor global view, however, CrossTalk has shown performance improvement in a load balancing algorithm specifically reducing per-hop packet delay and bottlenecks. It seems reasonable to expect such performance to be improved by improved global modeling of the network and this expectation is the encouragement for the new cross-layer architecture.

Researchers have addressed the importance of "knowledge plane" or "knowledge network" for the adaptability or context awareness in next generation communications. Clark *et al* [13] describe the Knowledge Plane as a new communication paradigm where a network can assemble itself given a set of high-level instructions and constraints, re-assemble itself as requirements change, automatically discover when something goes wrong, and automatically fix a detected problem or explain why it cannot do so. This AI-based approach considers an additional network layer between the network and the application layer, as the place in which nearly all network control activities take place. Mulvenna and Zambonelli [14] present an abstract architecture for knowledge networks that addresses the key issues of how both physical contextual knowledge and social knowledge from the users of communication networks can be used to form a knowledge space in support of autonomic agents dealing with network elements and applications. To avoid the burden of an additional distributed computational layer, and to more successfully promote cross-layer interactions, they consider knowledge networks as managed by existing components at the application and network levels. Both proposals are at the abstract level: still, these are inspiring issues for the use of a Knowledge Plane in our architecture. However, our cross-layer, architecture-oriented knowledge plane approach is somewhat different than these approaches.

It is possible to utilize cross-layer information to attain some of the self-behaviors of autonomic communications mentioned earlier. POEM (Performance-Oriented Model) [10] is perhaps the first initiative towards developing a cross-layer based self-optimizing protocol stack specifically for autonomic communication. For the optimization purposes it utilizes local state information. The basic design criterion is self-optimization is a control-plane issue where the normal functions of the protocol stack should not be compromised, with added cross-layer benefits being layered on top. The system is being investigated both formally and through simulation. In their patent filing, Sadler *et al* [15] link cross-layer information to self-healing. In this architecture, a Management Plane exists along

side of the protocol stack to stores information obtained from each protocol layer in the Cross-Layer Platform and nodes use this information to help the routing protocol maintain network reliability in the presence of failures. Interestingly none of the self-* behaviors in autonomic computing and communication are highly orthogonal, which means there is some dependency between them – self-healing is partly supporting self-organization, and vice versa.

3 A Cross-Layer Architecture Based On a Global View

Both OSI and TCP/IP adopt a bottom-up approach driven by physical and network constraints, which can make it hard to capture and respond to top-down user demands or contexts. Cross-layer design can help capture these concerns by providing a more uniform framework within which to capture and disseminate information at different semantic levels. Realizing the importance of cross-layering – and specifically cross-layering architectures with a network-wide global view – in next-generation communications, we propose an alternative cross-layering architecture based on a combination of node-local and network-global views. “Global view” is a broad term suggestive of centralization and reduced scalability, but we will try to establish and maintain a network-wide view of multiple metrics depending on the focus of the network without undue impact. Metrics such as load, battery status and so on in can be very useful in attaining self-organization or self-healing in an autonomic network, and having a global view allows a node to evaluate its own status against the average within the network at any instant. For example, a node could decide whether it is carrying more traffic then the average node and how overloaded it is compared to the average, and could utilize this information for routing and load balancing.

As this architecture is based on local information as well as global and the sources of information are different, we require two different schemes to gather information. This leads to the use of a knowledge plane consisting of two entities to manage information efficiently. One entity is responsible for the organization of locally available information from different layers in the local node’s network stack, provided by each layer of the protocol stack independently: battery status, load, neighbor count, signal-to-noise ratio transmit power, bit error rate, velocity and so forth. Each protocol layer can access this data and use it for local optimizations. The sum of this information represents the state of the node or *local view* on the network. The other data-management entity establishes a network-wide or *global view* summarizing information of the same types collected in the local view. To produce the global view we use an information dissemination procedure based on gossiping.

3.1 Motivation of this Architecture

Why another architecture? We believe that existing approaches contain some excellent approaches to particular problems in the construction of autonomic networks with cross-layer optimization: however, there is also only a fairly weak integration of the various techniques into a systems-level view of cross-layer interactions.

Cross-layer interactions can be managed by information exchange between layers, or through a more structured knowledge plane approach. Per-layer interactions are potentially more efficient, but also expose significantly more design information to the individual layers, which can lead to unnecessary coupling between implementations. It also tends to bind information to its source, in the sense that a layer will always have to commit to the layer originating information it requires for optimization. A knowledge plane decouples information from its source, abstracts the internal designs of layers and provides a central facility within which to provide reasoning and other advanced features.

Self-* properties are almost always phrased in whole-network terms: self-optimization, for example only makes sense if the network has a global property to optimize against, as individual local optimizations are unlikely in general to converge to a global optimum. A naïve implementation of a knowledge plane would inevitably be a performance bottleneck. Maintaining a global view without introducing performance and reliability concerns implies constructing the global view in a distributed manner, accepting the inevitable problems with latency and inconsistency.

Maintaining a distributed representation of knowledge means either locating some of the information at a particular node, or replicating the entire knowledge plane on each node, or a combination of the two. Since we do not want to over-commit to a particular strategy, we use a gossiping approach coupled with information summarization and fusion to maintain local copies of the knowledge plane. The local component acts as a gateway to the complete knowledge base, containing both local and summarized global views important metrics. A client layer need not know the source of particular information, and tailoring the gossiping algorithm provides a single point for handling robustness and latency issues. Using randomized selection of nodes with which to gossip, for example, produces a random communication structure that behaves rather like noise and does not introduce hot-spots or other artifacts. We conjecture that the overheads of such a scheme will be greater than the piggy-backing used in CrossTalk but significantly less than in deterministic flooding [16].

3.2 Overview of the Architecture

Alongside the existing layers, the Knowledge Plane is the key element of the architecture. Direct communication between layers and a shared knowledge plane across the layers are the two widely used cross-layer interactions polices. Because of the improved separation and management possibilities we prefer to utilize the knowledge plane for the architecture. The following are the main elements of the architecture:

Existing TCP/IP layers: These provide normal layering support when it is necessary, as well the information to the knowledge plane related to different layers to maintain local view of the node. This allows full compatibility with standards and maintains the benefits of a modular architecture, as it does not modify each layer's core functionality.

Contextors for different layers: Each layer in the existing protocol stack has a corresponding *contextor*, which will act as their corresponding interface between the layer and the knowledge plane. Each of these contextors acts as a "tuner" between a layer and the knowledge plane. Possible functionality for manipulating protocol data structures is built into the contextors: no modification is required to the existing protocol stack. This facilitates incorporation of new cross layer feedback algorithms with minimum intrusion. Generally a protocol implementation has data structures for control and data, with a protocol's behavior being determined by its control data structure. A contextor is responsible for reading and updating the control data structures when it is necessary.

Knowledge Plane: A common knowledge plane database is maintained to encapsulate all the layers' independent information as well as the network-wide global view, which can be accessed by all layers as needed. For modularity it maintains two entities responsible for maintaining the local and global views.

A knowledge plane can act in one of two ways: as a simple database with local and global view related information, or as a database with intelligence, which can manipulate its information to make inferences. The former allows each layer to provide whatever querying and optimization functions it requires, but forces all the complexity for information fusion into the contextors; the latter allows certain fusion and uncertain-reasoning functions to be encapsulated within the knowledge plane, making them available uniformly to the contextors and simplifying their local coding – at a cost of complicating the knowledge plane with the consequent risks of performance problems and feature interaction. There is clearly a trade-off in functionality that requires careful exploration.

Interaction between different layers and the knowledge plane through client programs can be reactive (responding to changing context) or proactive (anticipating changes and provisioning accordingly). Generally the interactions between different layers and the knowledge plane are event-oriented, which suggests a reactive scheme; on the other hand, the knowledge plane can maintain a model of the network and act autonomously to issue its own events. This leads to improved performance if the model leads to a correct proactive adaptation, but can be detrimental if the projection is wrong.

In our architecture we are considering the database with intelligence as shown in figure 1. The knowledge plane consists of the database and necessary optimizing algorithms. The database is separated into local view and global view for isolation and management purposes, although it appears unified to clients.

Gossiping: Gossiping is considered as one of the most promising data-dissemination mechanisms in peer-to-peer or distributed systems. There are number of algorithms that can be classified as *reactive*, *proactive* and *periodic*. As there are few comparative performance studies amongst these algorithms, it is difficult choose the most suitable

algorithm for a particular purpose. In our case we propose a periodic gossiping approach, possibly with out-of-band "immediate" signaling for important changes.

The gossiping service is built on top of existing TCP/UDP, and is responsible for gathering information from other nodes to generate the network-wide view at the host node. At each exchange the gossiping service chooses another node in the system (either randomly or with some weighted preference) and exchanges its local state with that node. In this architecture we will consider a gossiping exchange as an application-level event which will trigger the knowledge plane to take the necessary actions.

3.3 Interactions between Protocol Layers and the Knowledge Plane

Events and the state variables of different layers are the two most important concerns for the interaction between the layers and the knowledge plane. This interaction is the most delicate issue in a cross-layer architecture. There are two basic models:

- The knowledge plane registers with the contextors, and receives events whenever a change in the layer's context occurs; or
- The knowledge plane periodically retrieves state information form all layers.

We will here consider only the first approach further, as we believe it is better-suited to autonomic systems. The interaction policy between the layers and the knowledge plane through the contextors is summarized below (and as shown in figure 1):

Step 1: A contextor attaches to the control data structure information of the appropriate layer.

Step 2: The database part of the knowledge plane registers with the contextors to receive change events.

Step 3: Contextors notify the knowledge plane's database about events (timeout, disconnections, *etc*) and pass the necessary information regarding these context changes, which are incorporated into the model. Optimization algorithms monitor the database waiting for "guard" predicates to become true, and are executed when their guards are satisfied.

Step 4: The algorithms access the model to acquire any additional information they need in order to make their decisions. Algorithms are not restricted in which parts of the model they may access.

Step 5: Control actions are propagated back through the contextors to the layers' control plane.

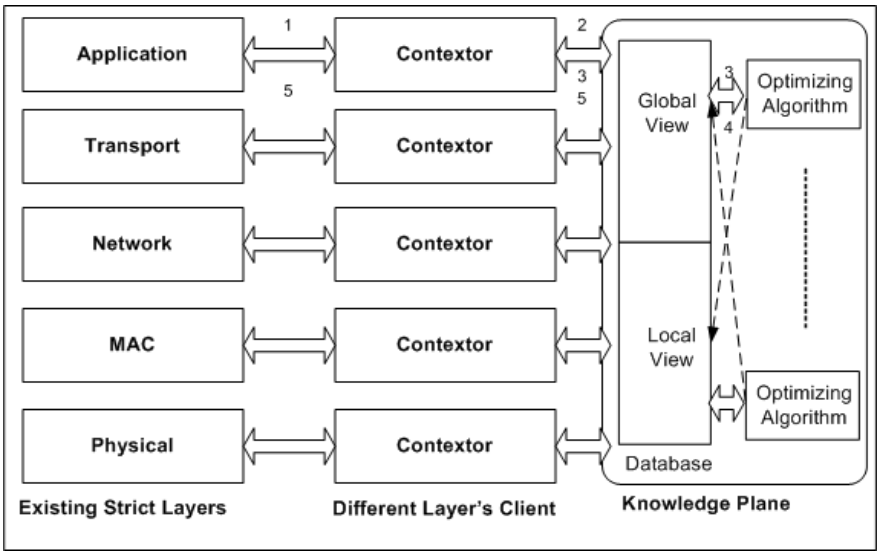

Fig. 1. Sample interactions between different layers and the Knowledge Plane

3.4 Application Scenario

Prospective application areas for this architecture could be optimization of load balancing, routing, energy management in wireless networking and obtaining self-behaving properties of autonomic communications like self-organization. All of these applications require some knowledge about their neighbors, which can be provided by our architecture.

As an application scenario, consider the case of a wireless network self-organizing to maintain communications between nodes. Self-organization can be defined as the emergence of system-wide adaptive structure and functionality from simple local interactions between individual entities [17]. An application scenario with 7 nodes is shown in figure 2. In scenario (a) node s has a request for node d1 and it is using the route s-n1-n2-d1. In this case, we will consider all the nodes are gossiping knowledge (the global view) about their primary neighbors, so node s has knowledge about n1, n1 has about n2 and n3 and so on. If after transmission begins n2 fails, existing (local-view only) routing protocols would have n2 receiving the packet, determining d1 to be dead, and finally sending a "node unreachable" error message to s which wastes all the resources committed to the exchange. Using a global view, however, if d1 and d2 are giving almost same type of services a suitable global view would allow n2 to determine that in case of d1's failure d2 can meet the request of s. This requires making information about the service-level capabilities of a node available to the routing layer, which is

facilitated by our approach and can easily be expressed as an optimization algorithm. This leads to scenario (b) where nodes have re-organized because of the death of d1, and once n2 gets the request from s it reroutes to d2 instead of d1 and fulfills the request. With this action, our architecture can conserves energy and minimizes latency by eliminating the overhead required to invalidate the current route, establish a new route, and retransmit the request. Moreover, it can preserve the original route when failed node becomes available. Essentially we provide in this scenario a generalized form of content- or service-based routing, without committing extra specific resources or network topology to the task and so retaining complete freedom to adopt other strategies as required.

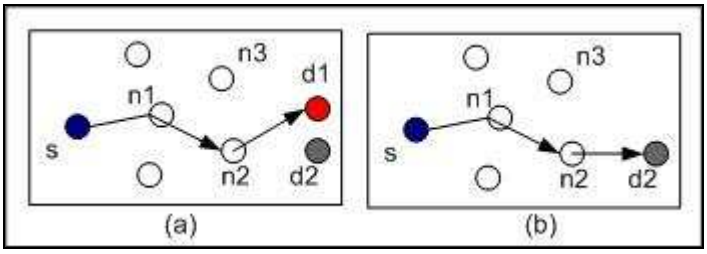

Fig. 2. Application Scenario

4 The Difficulties Implicit in Cross-Layer Interactions

Cross-layer design is not an easy task, as co-operation among multiple protocol layers has to be coordinated without leading to conflicts or (worse) loops. A common drawback is the lack of a systematic approach for cross-layer designs overall, not just its interactions: individual optimizations run at cross purposes, violating the structural architecture principles for only shortsighted performance gains, and could lead to serious consequences through unexpected feature interactions.

Once layering is broken, the luxury of designing a protocol in isolation is lost too. Unbridled cross-layer interactions can create loops, and from a control theory point of view become a hazard to the stability of the system. Loosely controlled interactions can also result in "spaghetti code," stifling further innovation and proliferation on the one hand, and increasing the cost for maintenance on the other. In severe case, the overall system will have to be redesigned. (These problems are detailed in [11] with examples.) At a critical time in the history of wireless communication, we need to note some of the adverse possibilities and exercise proper caution. Cross-layer design for new network architecture is the trend; however we think there are a few principles that have to be followed:

- Cross-layer design should be holistic instead of being fragmenting

- Layering should remain, for proliferation and longevity of the system
- Cross-layer interactions should be done in a controlled way, and preferably through a common optimization interaction interface
- Dependencies between the interaction protocol parameter have to be examined to avoid loops

Cross layering tries to share information amongst different layers, which can be used as input for algorithms, for decision processes, for computations, and adaptations. This process of sharing has to be coordinated and structured somehow since cross layering could potentially worsen the performance problem that it intends to solve. This is due to several effects. Optimization processes at different layers could try to optimize a common metric in opposite directions. Furthermore, two different metrics could have negative impacts on each other when trying to optimize them, such as energy efficiency and delay. A general problem is that altering a metric at one layer often has an effect on other layers implicitly. For example, altering the transmission power on the physical layer can have an effect on the network layer as nodes might disappear from the direct transmission range.

5 Conclusions and Future Work

The worldwide success of the Internet has led to the domination of the layered architecture, but a strict layered design is not flexible enough to cope with the dynamics of next-generation communications. Moreover exploitation of some cross-layer protocol interaction can lead to more efficient performance of the transmission stack (and hence better application layer performances) in different wireless networking scenarios. Most existing architectures depend on the local information and only CrossTalk depends on local as well as network wide global view. Even with the uncertain and poor global view gathering process, CrossTalk has shown performance improvement in load balancing algorithm. With a more accurate global view gathering process we can hope for further improvements, and with this in mind we proposed an alternative cross-layering architecture for autonomic communications which is based on local information as well as global information and gossiping will be the mechanism to collect the global view related information. The use of gossiping provides more impact that piggy-backing but keeps overheads more controlled than flooding, and allows the time constants and latencies of information to be varied widely.

Our proposed architecture is still in the design stage. We are currently engaged in exploring how we may provide cross-layer optimization of TCP/IP, simulating existing TCP enhancements and comparing them to a novel version whose adaptations are driven from a knowledge plane populated using our gossiping technique. This will allow us to evaluate both the gossiping approach and the various cross-layer parameters that can be used to influence TCP behavior, as well as comparing them against established approaches with less cross-layer influence.

References

1. Srivastava, V. and Motani, M, “Cross-Layer Design: A Survey and The Road Ahead”, IEEE Communications Magazine, Volume 43, Issue 12, Page(s): 112 – 119, Dec. 2005
2. Simon Dobson, “Putting meaning into the network: some semantic issues for the design of autonomic communications systems”, In Mikhail Smirnov, editor, Proceedings of the 1st IFIP Workshop on Autonomic Communications, volume 3457 of LNCS, Springer Verlag, 2005,pp: 207-216
3. H. Balakrishnan, “Challenges to reliable data transport over heterogeneous wireless networks”, Ph.D. Dissertation, The University of California at Berkeley, USA, 1998
4. ‘Autonomic Communication Forum’, HYPERLINK "http://www.autonomic-communication-forum.org" http://www.autonomic-communication-forum.org
5. ‘Autonomic Communications Initiatives’ HYPERLINK "http://www.autonomic communication.org" www.autonomic-communication.org
6. Daniel G.Sachs et al, “GRACE: A Hierarchical Adaptation Framework for Saving Energy”, ACEED 2005
7. Dzmitry Kliazovich and Fabrizio Granelli, “A Cross-layer Scheme for TCP Performance Improvement in Wireless LANs”, Globecom 2004, IEEE Communications Society, pp. 841-844
8. Marco Conti, Gaia Maselli, Giovanni Turi, Sylvia Giordano “Cross layering in mobile Ad Hoc Network Design”, IEEE Computer Society, pages 48-51, February 2004
9. Winter, R.; Schiller, S.; Nikaein, N.; Bonnet C., “CrossTalk: A Data Dissemination-based Cross-layer Architecture for Mobile Ad-hoc Networks”, IEEE Workshop on Applications and Services in Wireless Networks (ASWN 2005), Paris, June 2005
10. X. Gu, X. Fu.H. Tshofenig, and L. Wolf, “Towards Self-optimizing Protocol Stack for Autonomic Communication: Initial Experience”, Proceedings of the 2nd IFIP International Workshop on Autonomic Communication (WAC'05), Springer Lecture Notes in Computer Science Vol. 3854 (LNCS), October, 2005, pp: 186-201
11. V. Kawadia and P.R. Kumar, “A Cautionary Perspective on Cross-Layer Design”, IEEE Wireless Communications, February, 2005, pp.3-11
12. V. T. Raisinghani and Sridhar Iyer, “ECLAIR: An Efficient Cross Layer Architecture for Wireless Protocol Stacks”, WWC2004
13. D. Clark et al., "A Knowledge Plane for the Internet", Proceedings of the 2003 ACM SIGCOMM Conference, Karlsruhe (D), ACM Press, 2003
14. M. Mulvenna, F. Zambonelli, "Knowledge Networks: the Nervous System of an Autonomic Communication Infrastructure", 2nd IFIP Workshop on Autonomic Communication, LNCS No. 3854, 2006
15. C. Sadler, W. Chen, and L. Kant. , “Cross-Layer Self-Healing in a Wireless Ad-Hoc Network”, U.S. Patent filed April 2005
16. M. Lin, K. Marzullo, S. Masini, "Gossip versus deterministic flooding: Low message overhead and high reliability for broadcasting on small networks", UCSD Technical Report TR CS99-0637
17. Prehofer, C.; Bettstetter, C.**, “**Self-organization in communication networks: principles and design paradigms”, IEEE Communications Magazine, July 2005 Page(s): 78 - 85