Policy-Based Quality of Service and Security Management for Multimedia Services on IP Networks in the RTIPA* Project

ValÈrie Gay[1], Sandrine Duflos[1], Brigitte Kervella[1], Gladys Diaz[2], and Eric Horlait[1]

[1] LIP 6 : Laboratoire d'Informatique de Paris 6,
8, Rue du capitaine Scott, 75015 Paris, France
{Valerie.Gay, Sandrine.Duflos, Brigitte.Kervella, Eric.Horlait}@lip6.fr

[2] L2TI ñ Institut GalilÈe ñ UniversitÈ Paris 13,
99 Av. J. B. ClÈment, 93430 Villetaneuse - France
Gladys.Diaz@galilee.univ-paris13.fr

Abstract. This paper summarizes the research work that has been conducted in the context of the RTIPA project on policy-based QoS (Quality of Service) and security management for distributed multimedia services. It presents an architecture allowing the derivation of policies from the service level down to the network level. It is a step towards an end-to-end QoS and security management for distributed multimedia services running on the new generation of IP networks.

1 Introduction

Enabling end-to-end QoS and security management of distributed services is a research topic that has been going on for some years now. Services are getting adaptive and QoS-aware and the network is now able to deal with different levels of QoS and security. However, there is still a gap between the services and the network, and achieving end-to-end QoS and security for distributed applications is still bringing some challenges to the middleware and network research communities.

In the framework of the RTIPA project [1], our role was to reduce that gap and to make a step towards end-to-end QoS and Security management for distributed multimedia services running on the new generation of IP networks.

In the project we dealt with policy-based network management therefore we naturally focused our research on a policy-based solution. The major part of current standardization works on policy-based management is done at the Internet Engineering Task Force (IETF) and are positioned at the network level. The IETF proposes a centralized management [2]. This model is composed of several entities: the Policy Repository (PR) where policies are stored, the Policy Decision Point (PDP)

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-540-45812-8_28

* RTIPA: Real-Time Internet Platform Architectures, ITEA (Information Technology for European Advancement) project

K.C. Almeroth and M. Hasan (Eds.): MMNS 2002, LNCS 2496, pp. 25-35, 2002.

that decides the policies to apply, and Policy Enforcement Points (PEP) that enforce the policies. At the network level, IETF also defined an information model: The Policy Core Information Model (PCIM) [3] [4]. It has also defined extension for QoS management: The QoS Policy Information Model (QPIM) and the QoS Device Data path Information Model (QDDIM) [5] [6]. For security management, the IPsec Configuration Policy Model (ICPM) [7] describes the information for policy management. Other research works propose solutions to specify, manage and enforce policies at network or middleware level. Examples are found in [8] that describes a solution for IPsec policies, [9] that bases its solution on the Corba Security framework [10] and [11] that proposes an object-oriented solution to manage and enforce policies expressed in a language called Ponder [12].

Our work presents an architecture for end-to-end QoS and security management of distributed multimedia services. It focuses on the top-down aspects of the policy management and its relationships with security and QoS managers. It also describes how it could be implemented on an Ipv6 network that takes into account the service level policies thanks to a refinement process.

This paper describes in Section 2 an architecture for policy-based management of QoS and security for multimedia services. Section 3 describes the policy refinement process. Section 4 presents how the policies are supported at the network level on the RTIPA platform. Section 5 concludes on open issues and perspectives

2 An Architecture for Policy-Based QoS and Security Management

Our architecture is presented in Fig. 1. It is separated into three abstraction levels: service, middleware and network. Fig.1 only highlights the elements and interactions we are discussing in this paper. Other managers that do not appear in the figure include the billing, mobility and Service Level Agreement (SLA) managers. At the service level, QoS, security, and policy rules are expressed in high-level terms understandable by the end-users. A QoS contract specifies the service requirements and offerings in terms of QoS and security. Its granularity can vary from 'per service' to 'per flow type'. The service level policy rules can express general administration rules (such as routing policy rules in the case of network management) in high-level terms. The policy manager maps them onto middleware policy rules. The service level QoS, policy and security requirements and offerings are translated into middleware level policy rules (using the middleware QoS, security and policy managers) to be integrated into the new set of middleware policy rules.

The middleware policy manager checks the policy rules and resolves the conflicts. Feedback can be provided to the application level and can lead, for example, to the re-negotiation of QoS or price or to the adaptation of the content.

The QoS and security managers check if the different QoS and security requirements specified at the service level can be met by the infrastructure and provide feedback to the application on the status of the infrastructure with respect of the application QoS and security. The resource manager [13] has a view of the available resources. It enables the QoS and security managers to know if their QoS

and security can be fulfilled. Especially for real-time service it is important to have strict resource management policies so the real-time QoS requirements are valid during the lifetime of the service.

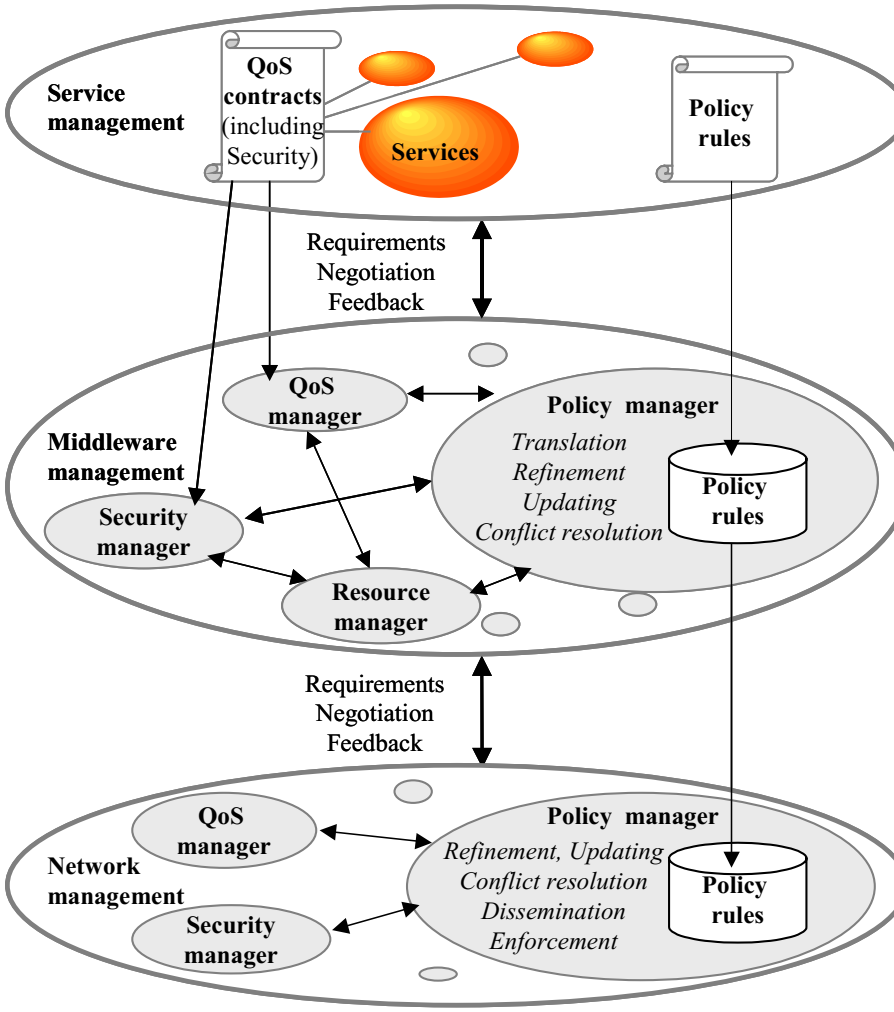

Fig. 1. Policy-Based QoS and Security Management Architecture

The middleware level policy rules are then translated into network-level policy rules. The network policy manager adapts the rules to reflect the requirements it gets from the other network managers (such as a network QoS manager). It is disseminating the rules and enforcing the policies. It resolves the conflicts between the policy rules and provides feedback to the middleware managers.

QoS management and mapping have been discussed in our previous work [14]. In that paper, the content of the QoS contracts are detailed, they express the QoS

requirements and obligations of the distributed services. The QoS manager [13] monitors the network QoS and can initiate operations like QoS re-negotiation in case of QoS violation. It checks if the environment can and will fulfill the contracts. It requests the resource manager for certain resources (e.g. buffer, CPU-time, Audio-device) that can be allocated to a certain service. There is a need of cooperation of QoS managers in different domains.

The goal of the security manager is to provide the necessary security that is required by the multimedia service. The security manager analyses and translates these requirements, expressed in the QoS contract at service level, in security parameters depicting the security aspects to provide. For example, in the case of a Video-on-Demand (VoD) service, a minimum of security is necessary to protect the audio/video streams against a fraudulent listening. In such a case confidentiality is necessary. The resource manager is checked with to see if a security protocol can set this security aspect. If yes, it sends the parameters to the policy manager that integrates them in the middleware level policy rule. In order to produce the network level policy rules, the network level security manager must be able to provide the security protocols and cryptographic algorithms available to set the security required. In our VoD example, the confidentiality parameter becomes ESP and (3DES, DES) ESP being the IPsec protocol providing confidentiality and 3DES and DES a proposition of associated cryptographic algorithms. The security and policy managers interact a last time for the protocol configuration, in our example IPsec, when the network policy rule is disseminated.

In case the policy managers cannot resolve the conflicts, the security managers must be able to provide an alternative solution (e.g.: security parameters re-negotiation at middleware level or proposition of others protocols and cryptographic algorithms at network level) to produce a new policy and to secure the service. If it is not possible the service should be turned off.

The managers shown in Fig. 1 would typically be located on every node and cooperate among each other to manage the distributed multimedia service. We are not describing their internal mechanisms in this paper.

Policy rules [15] evolve through a life cycle (specification, refinement, static and dynamic conflict resolution, dissemination, enforcement, updating (includes disabling, withdrawing and deleting). Actions related to this life cycle are shown in Italics in Fig. 1. In the following section, we focus on the refinement process.

3 The Policy Refinement Process

This section describes the policy translation and refinement process from the middleware to the network levels. Our work is based on the TMN (Telecommunications Management Network) architecture defined by the ITU-TS [16] and the research work of [17]. The different managers presented at middleware level derive from the service level the rules to be generated and included in the middleware-level policy rules. A possible template for the middleware level policy is described in Table 1 where <*ConfigurationType*> is the network configuration type (e.g.: point to point, multicast, star, fully meshed.), <*TransportService*> is the

requested transport service (e.g., ATM, SDH/PDH, IP), <*ApplicationType*> is the requested application (real player, Netscape, video on demand), <*SourceUser*|*user..**> and <*DestinationUser..**> are the identified users who will participate to the application. <*QualityType*> is the requested QoS level (e.g.: premium, gold, silver, bronze). <*SecurityConfiguration*> is the requested security configuration including confidentiality, authenticity (integrity and authentication of packet sender), mutual authentication, non-repudiation, tunnel mode and anti-replay.

The different parameters values are checked by the middleware managers described in the previous section (e.g.: are the recorded members known by the service? is the requested security configuration correct?)

Once the request is validated, it can be translated into a middleware-level policy rulerepresented here under the form "If condition Then provide configuration" that will support the implementation of the QoS and security requested.

Table 1. Middleware level policy

IF SourceUser|users = <*SourceUser*|*user..**> AND DestinationUser = <*DestinationUser..* (Optional)* > AND ApplicationType = <*ApplicationType*> THEN PROVIDE <*ConfigurationType*> of <*TransportService*> for <*ApplicationType*> with Guarantee <*QualityType*> Quality and <*SecurityConfiguration*> Security

The middleware level policies are only understandable at that level. They use middleware-level parameters such as a specific security configuration (such as confidentiality) or the users' name. Some parameters do not have to be managed by the network because they are dealt with at a higher level but the rest of the parameters have to be converted in network level parameters. In the RTIPA project, our network is based on IPv6 therefore we are using IP Diffserv and IPsec [18] [19] [20] [21] at network level. A simplified translation table is presented in Table 2

Table 2. Simplified Translation from Middleware to Network-Level Parameters

	Middleware level parameters	Network level parameters
User	<*Users*> (e.g.: End-User1)	<*UserIpAddress*> (e.g.: 2.2.2.2)
Application	<*ApplicationType*> (e.g.:VoD)	<*PortNo*> <*QoSDirection*> <*ConnectionType*> <*InterfaceIPAddress*> (e.g.: 8000, uni, unicast, 2.2.0.0))
Configuration and transport	<*ConfigurationType*> and <*transport type*> (e.g. point to point, IP)	Identical
QoS	<*Quality Type*> (e.g.: gold)	<*PhBType*> (e.g.: AF11)
Security	<*SecurityConfiguration*> (e.g.: confidentiality)	<*Sec-prot*>, <*C-Algo*>, <*A-Algo*>, <*Mode*>, <*No-Replay*> (e.g.: ESP, (3DES, DES), NULL, IPsec_transport, NULL)

In Table 2 and Table 3, <*UserIpAddress*> corresponds to the users IP addresses, <*PortNo*> is the port used, <*QoSDirection*> is the QoS direction (unidirectional or bi-directional) the <*ConnectionType*> is the connection type (e.g. unicast, broadcast, multicast) and <*PhBType*> is the desired Per-Hop behavior. <*Sec-Prot*> is the IPsec security protocol used (AH or ESP), <*C-Algo*> is a list of algorithms used for confidentiality (e.g.: {private, key cryptography} or NULL), <*A-Algo*> is a list of algorithms used for authenticity (e.g.: {digest function, public key cryptography} or signature cryptography or NULL), <*Mode*> is the security mode (tunnel or transport) and <*No-Replay*> is set to True or False to indicate that it can or cannot be replayed.

For instance, the security parameters, expressed at the middleware level, will be replaced by the IPsec security protocol (Sec-Prot), a list of algorithms used for confidentiality (C-Algo), a list of algorithms used for authenticity (A-Algo), the security mode (tunnel or transport) used for the creation of secure tunnel (e.g. VPN) and the anti-replay protection (No-Replay). Furthermore, a user will be replaced by its IP address, the QoS type by the corresponding DiffServ PHB (Per Hop Behavior) type, etc. This modification of parameters allows the co-ordination of the policy implementation in the whole network. The new policy rule template is shown in Table 3.

Table 3. Network-Level Policy

IF SourceIPaddress|UserIPaddresses = <*SourceIPaddress*|*UserIPaddresses1..**> and SourcePortNo|UserportNo=<*SourcePortNo*|*UserportNo*>and DestinationIPAddress = <*DestinationIPAddress..(optional)*>and DestinationPortNo =<*DestinationPortNo (optional)*> THEN CONNECT with <*QoSDirection*> and <ConnectionType> from|among <*SourceIPAddress!..**> at <*SourcePortNo*|*UserPortNo*> to <*destinationIPAddress!..*(optional)*> at <*DestinationPortNo1 (optional)*> with <*PhBtype*> and <*Sec-Prot*>with <*C-Algo*> and <*A-Algo*> and <*Mode*> and <*No-Replay*>

A last step is necessary to disseminate and enforce the policies rules in the different network entities. Only the rules and parameters of concern for the entity where the policy will be enforced are disseminated.

Table 4. Network-Level Policy for Dissemination to the Network Elements

IF SourceIPaddress|UserIPaddresses = <*SourceIPaddress*|*UserIPaddresses1..**> and SourcePortNo|UserportNo = <*SourcePortNo*|*UserportNo* >and DestinationIPAddress = <*DestinationIPAddress..(optional)*> and DestinationPortNo = <*DestinationPortNo (optional)*> THEN SET at <*InterfaceIpaddress*> With <*PhBtype*> and <*Sec-Prot*> with <*C-Algo*> and <*A-Algo*> and <*Mode*> and <*Anti-Replay*>

For example for a videoconferencing service, when two users from distinct domains (managed by distinct entities) communicate, each entity receives the IP address of the other user. This policy rule allows the definition and the establishment of security associations between the communicating entities to ensure an end-to-end security. Table 4 presents the policy that will be implemented in the network elements. The keyword SET makes it clear that the objective is no longer to provide

something but to set something to certain values. The <*interfaceIPAddress*> represents where the policy must be enforced. It generally represents an edge router on which a set of <*UserIPAddress*> is connected.

This is a first step towards automatic translation. Ideally, the refinement process could be automated through the use of a translation table as depicted in Table 2 but the translation process is more complex than that. In addition there is a need to resolve conflicts and to deal with the fact that some resources might not be available.

4 Network Level Prototyping

In the RTIPA project, we worked on different aspects of the architecture. One of our task was to study languages mapping between applications using SMIL [22][23] and the middleware level. However, the network level was the only really stable basis for the implementation. Standards on network policy management exist and the choice of technology had been made [24]. It is therefore on that level that we chose to demonstrate some aspects of our architecture. Our objective was to integrate QoS and security policies using Traffic Designer, a network-level tool developed by Qosmos [25].

This section describes how Traffic Designer is used to support policy-based QoS and security management at the edge routers level. The architecture of the Traffic Designer (TD) and its interactions with the different managers of our architecture are depicted in Fig. 2.

Traffic designer is an integrated tool able to provide actions on the flows (classification, prioritization, throughput guarantee, forbiddance, tagging of packets, counting, analyzing and spying) based on a set of rules. This set of (policy) rules integrates the policy rules that have been derived as described in section 3 (Table 4).

Traffic designer is composed of a classifier (*TD classifier*), a filter (*TD filter*) and five action modules (*TD QoS*, *TD Firewall*, *TD cryptography*, *TD Diffserv* and *TD Monitor* and its related *TD log database*).

The *TD classifier* takes as input packets or packet flows and gives as an output the highest level of protocol the packets belongs to. This allows us to provide rules on those particular applications.

The *TD filter* applies the rules (including our policy rules) and redirects the packets to the action modules if appropriate. Examples of rules are:

- IF (udp_dport=1024..65535) AND (tcp_dport=1024..65535) Then goto Firewall (rule based on the header fields)
- IF (http_mime=" image/gif ") AND (smtp_attach=" application/pdf") Then provide 20 Kbps (rule based on the application properties)
- IF (flow_content=*topsecret*) then goto Firewall (rule based on the packet content (not advised))
- IF (client to server) then...(rule based on the direction of a TCP connection).

The *TD QoS* module is able to shape the traffic, to apply queuing mechanisms and to recover the unused bandwidth. It gives the possibility to guarantee a minimum or a maximum throughput, to tune the maximum waiting delay for the packets, to provide

priorities within a queue (e.g. LIFO). The packets can then be tagged depending on applicative criteria. It enable us, in particular, to guarantee bandwidth to mission critical applications and establish classes of traffic depending on the protocol used and the user involved.

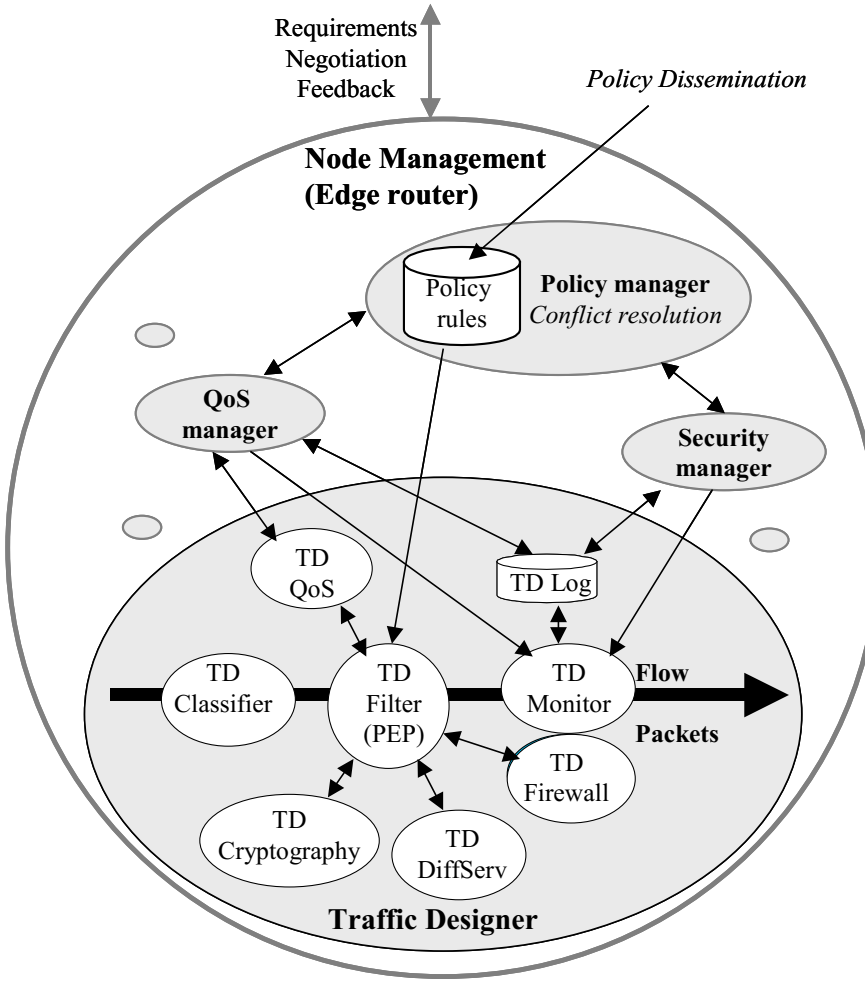

Fig. 2. Use of Traffic Designer at the Edge Routers

The *TD firewall* (security) can drop packets. It provides us with functionalities to protect the application against hackers, to filter per protocol or per application URL and keywords, to position alarms on certain event. This security aspect is interesting but not sufficient for our security needs.

To enforce our security policy rules we need another action module, the *TD cryptography* module that is currently under development. This new module will provide a user-transparent cryptography (multiple key system, choice of the key depending on the application or of the user). When available, this feature will be

studied in order to know if it can be used for the cryptographic algorithm management.

The *TD Diffserv* module adapts the flow to DiffServ.

The *TD Monitor* provides information and statistics in its *TD log database* (number of packets/bytes, instant/average/peak throughput, percentage of bandwidth used, zoom on a particular traffic type and spying, discovery based on applicative criteria). The syntax used to query the database is a simplified subset of SQL. Some examples are:

- SELECT (tcp_sport, ip_saddr)
  FILTER (tcp_proto=http) AND NOT (ip_saddr=" webserver ")
  Result:
  o 8080 192.168.2.5 79pkt, 10,5Kbps, 4342octets
  o 7979 192.168.2.8 4 pkt, 2,1 Kbps, 324 octets

- SELECT (smtp_sender)
  FILTER (smtp_subject=" *big brother*") OR (smtp_subject=" *loft story*")
  Result:
  o Anne.Aunime@qosmos.net 79 pkts, 0 Kbps, 4342 octets
  o Joel.Noyeux@qosmos.net 458 pkts, 2,1 Kbps, 33424 octets
  o Jean.Aymard@qosmos.net 2303 pkts, 0 Kbps, 23423 octets

The policy manager acts on the *TD filter* to enforce the policy rules. The QoS and security managers of the edge router interact with the *TD Monitor* to describe the information to track. They can then query the *TD Log database* to get the information they need and if necessary they can provide a feedback to the network, middleware and service levels.

5 Conclusion: Open Issues and Perspectives

This paper has presented our research work done in the context of the RTIPA project to manage QoS and security for distributed multimedia services running on IP networks. Some open issues still remain.

The services have a range of QoS requirements and offerings well identified. Various protocols and mechanisms exist to support QoS in distinct architectures. But the problem is how can we mix and match their capabilities to provide a complete QoS and security management? For example, at the network level, some incompatibilities appear between the IPsec and Diffserv protocols.

The refinement of high-level policies (service level policies in our architecture) is another issue: it is difficult to automate. It is likely to remain partially a manual process since it requires a large amount of knowledge. Also, what happens when the different rules cannot be validated (conflict between the rules...) or when the network cannot support the low-level policy? What sort of feedback should be provided to the user and will there be renegotiations of the QoS and security requirements and offerings? If it is the case, a relationship binding the different policy abstraction levels, as in [26] with the parent/child relationship is essential.

In this article, we have depicted a simplified version of QoS and security policy rules but a problem is observed during inter-domain communications. The domains

do not have necessarily the same way to represent policy, QoS and security. Several policy specification languages exist such as [27],[28],[29],[30],[31]. They provide flexibility and permit to express policies through different approaches (logical, graphical, etc.). Ponder [12] is the most suitable language for our solution. It integrates QoS and security policies, it is mature and an implementation is currently available. Another candidate for the security aspect would have been the Security Policy Specification Language (SPSL). There is a need for a global agreement on the policy language, and this not only at the network level, to be able to implement an end-to-end policy-based QoS and security management.

Our objective is to continue this research work and to extend this architecture to SLA and mobility management. The work on security is further detailed in another paper [32].

Acknowledgements

The authors would like to thank the members of the RTIPA project, Qosmos and the APS group of the university of Twente for their valuable comments on this work.

References

1. RTIPA http://www.extra.research.philips.com/euprojects/rtipa/ or http://www.itea-office.org/projects/facts_sheets/rtipa_fact_sheet.htm
2. R. Yavatkar et al., A Framework for Policy-Based Admission Control, RFC 2753, January 2000
3. B. Moore et al., Policy Core Information Model - Version 1 Specification, RFC 3060, February 2001. http://www.ietf.org/rfc/rfc3060.txt.
4. B. Moore et al., Policy Core Information Model Extensions V.7, http://www.ietf.org/internet-drafts/draft-ietf-policy-pcim-ext-07.txt, March. 2002.
5. Y. Snir et al., Policy QoS Information Model, Policy framework Working Group, Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-policy-qos-info-model-04.txt, November 2001.
6. B. Moore et al., Information Model for Describing Network Device QoS Datapath Mechanisms, Policy framework Working Group, Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-policy-qos-device-info-model-08.txt, May 2002.
7. J. Jason et al., IPsec Configuration Policy Model, Internet Protocol Security Policy Working Group, Internet Draft, February 2002.
8. J. Zao et al, Domain Based Internet Security Policy Management, Proceedings of DARPA Information Survivability Conference and EXposition 2000 (DISCEX '00), January 2000.
9. R. Chandramouli, Implementation of Multiple Access Control Policies within a CORBASEC Framework, 22th National Information Systems Security Conference NISSCí1999.
10. Corba "Security Service" Specification v 1.7, OMG, March 2001 (ftp://ftp.omg.org/pub/docs/formal/01-03-08.pdf)
11. N. Dulay et al., A Policy Deployment Model for the Ponder Language, 7th IFIP/IEEE International Symposium on Integrated Network Management (IM'2001), Seattle, USA May 2001.

12. N. Damianou et al., Ponder: Language for Specifying Security and Management Policies for Distributed Systems, The Language Specification Version 2.3, October 2000. http://www-dse.doc.ic.ac.uk/Research/policies/ponder/PonderSpec.pdf
13. P. Leydekkers et al., A Computational and Engineering View on Open Distributed Real-time Multimedia Exchange, In Lecture Notes in Computer Science - Number 1018 - Springer Verlag - ISBN: 3-540-60647-5 - Proceedings of Network and Operating System Support for Digital Audio and Video (NOSSDAV'95), Durham, New Hampshire, USA, April 1995.
14. P. Leydekkers and V.C.J. Gay, ODP View on Quality of Service for Open Distributed Multimedia Environments, In 'Quality of Service - Description, Modelling and Management', A. Vogel and J. de Meer editors. Proceedings of the 4th International IFIP Workshop on QoS, Paris, March 1996.
15. M. D. J. Cox and R. G. Davison, Concepts, Activities and Issues of Policy-Based Communications Management. BT Technology, Vol 17 No 3 July 1999.
16. Jajodia S. et al., Principles for a Telecommunications Management Network, ITU-T M.3010, February 2000.
17. H.J. Tjaja, SLA Enforced by Policy, masters thesis, KPN/Twente University internal report, 2001.
18. K. Nichols et al.., Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, RFC 2474, December 1998.
19. S. Kent and R. Atkinson, IP Authentication Header, RFC 2402, November 1998.
20. S. Kent and R. Atkinson, IP Encapsulating Security Payload (ESP), RFC 2406, November 1998.
21. S. Kent and R. Atkinson, Security Architecture for the Internet Protocol, RFC 2401, November 1998.
22. J.van Ossenbruggen et al., Towards Second and Third Generation Web-Based Multimedia. In: The Tenth International World Wide Web Conference, May 2001, Hong Kong
23. SMIL 2.0 documentation http://www.w3.org/TR/2000/WD-smil20-20000921/
24. M. Riguidel, Introduction to Policy Management, RTIPA Technical Report, 2000.
25. www.Qosmos.net
26. D. Marriott, PhD Thesis, Policy Service for Distributed Systems, Department of Computing, Imperial College, London, 1997.
27. N. Damianou, A Policy Framework for Management of Distributed Systems, PhD Thesis, Imperial College of London, February 2002.
28. T. Koch et al., Policy Definition Language for Automated Management of Distributed Systems, Proceedings of the Second International Workshop on System Management, IEEE Computer Society Press, 1996.
29. J. Hoagland, Specifying and Implementing Security Policies Using LaSCO, the Language for Security Constraints on Objects, PhD Thesis University of California, Davis Department of Computer Science, 2000.
30. M. Hitchens et al., Tower: A Language for Role Based Access Control, Policy 2001, LNCS 1995, pp.88-106, 2001.
31. Jajodia S. et al, A Logical Language for Expressing Authorizations. In Proceedings of the IEEE Symposium on Security and Privacy. Oakland, CA, USA: IEEE Press, 1997. p. 31-42.
32. S. Duflos et al., An Architecture for End-to-End Policy-Based Security Management, Submitted to the eighth IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), March 2003.