Automatic Dynamic Resource Management Architecture in Tactical Network Environments

Andy S. Peng*‡, Dennis M. Moen*, Tian He†, David J. Lilja‡

*Tactical Systems, Maritime Systems and Sensors (MS2), Lockheed Martin, Eagan, MN
†Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN
‡Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN
{andy.s.peng, dennis.moen}@lmco.com, tianhe@cs.umn.edu, lilja@umn.edu

***Abstract*—Automatic Dynamic Resource Management (AutoDRM) is a method to efficiently manage shared resources in the tactical network environments without human operator intervention. The AutoDRM architecture is developed as an attempt to resolve the resource contention issues and to improve the quality of service in the tactical network environments. The information herein describes the key components of the AutoDRM architecture. An experimental end-to-end network prototype test bed was also developed to host the AutoDRM system. Experimental results demonstrate improved network performance when AutoDRM is deployed.**

***Index Terms*—Performance Evaluation, Quality of Service, Resources Management, Tactical Network.**

I. Introduction

Providing End-To-End (ETE) Quality of Service (QoS) in the Department of Defense's (DoD) Global Information Grid (GIG) network is essential for supporting communication activities in tactical missions [1] [2]. An initial step towards such an ETE QoS support in the large-scaled network is to ensure that computing resources in each edge network domain are managed efficiently and in accordance with the GIG architectural framework [3] [4]. Future computing requirements for diverse tactical missions rapidly increase the complexity of the heterogeneous tactical edge networks such as the existing Total Ship Computing Environments (*TSCE*), upcoming Consolidated Afloat Networks and Enterprise Services (*CANES*) [5], Command and Control systems (*C2*), and Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance systems (*C4ISR*) [6]. Fig. 1 illustrates an operational view of tactical edge networks[1]. These tactical systems consist of many computing and networking devices highly integrated within a common network infrastructure in order to alleviate the number of tasks previously performed by human operators. The systems often provide real-time services such as Voice-over-IP (VoIP), streaming video, real-time messaging and other time-sensitive tactical applications that require stringent QoS guarantees with limited computing and networking resources. Each time-constrained application demands resources at different levels in order to achieve various QoS requirements. Without an adequate resource management solution, simultaneous increases in all QoS levels can result in resources contention which inevitably impacts the system performance. An automated method to dynamically allocate resources based on the prioritization across multi-dimensions (*e.g.* traffic classes, user precedence..*etc.*) and multi-choices (*e.g.* QoS policies..*etc.*) is needed to address such a technical challenge. The development of AutoDRM architecture mainly leverages the performance monitoring capability of a network management system (NMS) and policy-based QoS capability in the network domain. AutoDRM does not replace the overall network management system for the network domain. It merely serves as a supplemental function to the NMS by providing the capability for efficiently utilizing the resources within the context of a tactical edge network domain.

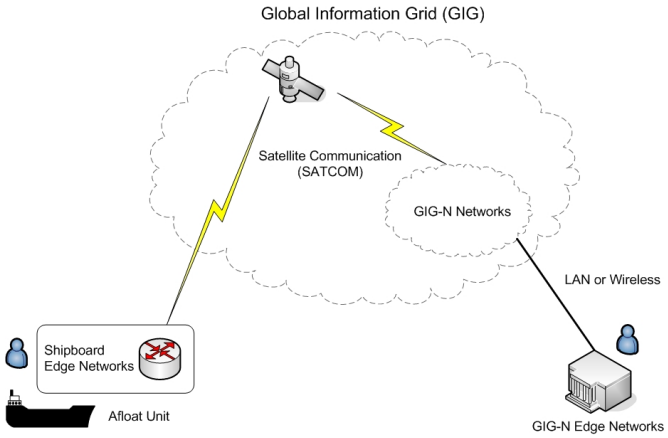


Fig. 1. Operational View of Tactical Edge Networks

A significant amount of research in dynamic resource management has been studied in various contexts. Rajkumar *et al.* [7] [8] developed a QoS-based resource allocation model (Q-RAM), which establishes an analytical approach to distribute system resources among multiple applications while maximizing the utility function. Harada *et al.* [9] had proposed an adaptive resource allocation method based on control theories. In the context of shipboard computing environments, Lardieri *et al.* [6] had developed a multi-layered resource management framework in enterprise distributed real-time and embedded (DRE) systems. They primarily focused on managing the

[1]GIG-N Networks refer to Tier 3 GIG networks (*e.g.* Defense Information Systems Network (*DISN*), Warfighter Information Network-Tactical (*WIN-T*)..*etc.*). GIG-N Edge Networks refer to user networks that connect through Tier 3 networks to the GIG black core [4].

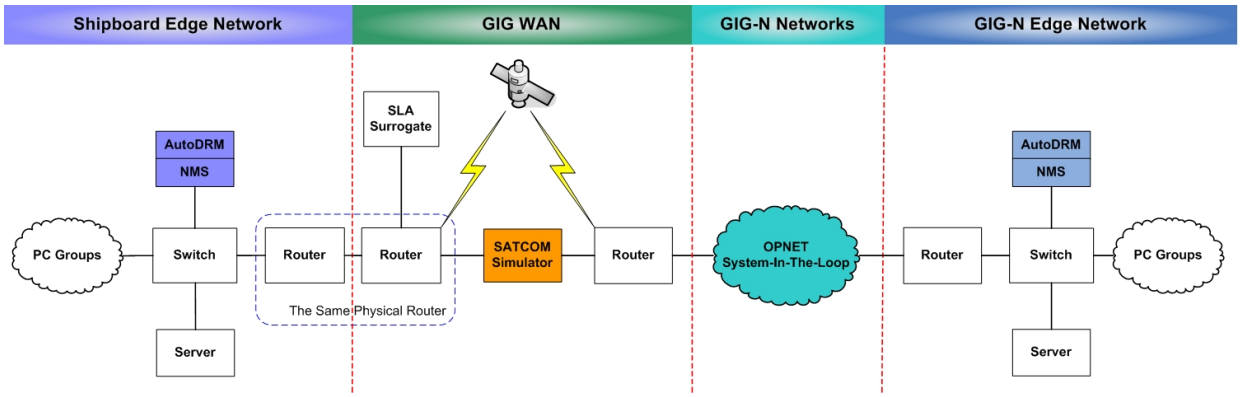


Fig. 2. End-To-End Network Prototype Test Bed

dynamics of computing resources in response to mission mode changes and/or resource load changes. Dasarathy *et al.* [10] developed a CORBA-based multi-layer management framework to manage changes in network resources, work load, and mission requirements at the network layer. Their study described the interactions of four key network QoS components: bandwidth broker, flow provisioner, network performance monitor, and network fault monitor. All of these studies in tactical network environments focused on providing a middleware framework to achieve QoS objectives. This paper focuses on the architectural concept of AutoDRM as well as the development of a network architecture prototype test bed to support the study.

The remaining portion of this paper is organized as follows. Section II reviews the theories related to the dynamic resource management problems. Section III provides an overview of the edge network architecture in the tactical network environments. An ETE tactical edge networks prototype test bed is also illustrated. Section IV presents the architectural concept of AutoDRM as well as details of each key functional component of the AutoDRM architecture. Section V describes a test scenario setup. SectionVI discusses the experimental test results. Finally, section VII concludes this study and discusses future works.

II. Theories

The dynamic resource management problem is generally formulated based on the 0-1 Knapsack problem which is known to be *NP-Hard* [11] [12]. Table I defines the general parameters of the resource management problem. A system provides n number of independent services (*e.g.* VoIP, streaming video, real-time messaging..*etc.*), $n \geq 1$. There are m number of shared resources (*e.g.* processing capacities, queue sizes, network bandwidth..*etc.*), $m \geq 1$. Each service S_i requires a set of shared resources r_j to accomplish its QoS objectives, where $i \in \{1..n\}$ and $j \in \{1..m\}$. A portion of resource r_j allocated to a service S_i is denoted by r_{ij}. Since each service often needs to meet a set of QoS requirements (*e.g.* packet latency, packet loss ratio..*etc.*), Q_{ij} represents a QoS requirement based on a service S_i consuming a shared resource r_j. This is done under the constraint that the total amount of resources is finite such that $\sum_{i=1}^{n} r_{ij} = R_j$ and $\sum_{j=1}^{m} R_j = \mathbf{R}$, where R_j is the maximum amount of each shared resource and $\mathbf{R}$ is the total available resources in the system. Since all of the resource requests may not necessarily be satisfied in a resource constrained environment, an actual achieved QoS level is represented by q_{ij} such that $q_{ij} \in Q_{ij}$. In order to accomplish adequate QoS levels for each service, the following condition must be met.

$$Maximize \sum_{j=1}^{m} x_i \cdot q_{ij} \quad subject\ to \sum_{j=1}^{m} x_i \cdot r_{ij} \leq R_j$$

$$where \ i = \{1, ..., n\} \quad and \quad x_i = \{0, 1\}$$

Fundamental theories in developing real-time near-optimal heuristics are discussed in [8] [9] [11] [12]. These approaches involve sorting orders in each data set and gradually assigning a portion of each shared resource r_j to each service S_i. If a QoS level q_{ij} satisfies the requirement Q_{ij}, the iterative process to assign resources is halted. Otherwise, it will continue to assign more more portion from each resource r_j to each service until the upper limit of that resource R_j has been reached. If a QoS requirement has not been satisfied after a specific resource R_j has been exhausted, a downgrade of resources is required from other services with lower priority. The priorities of services are determined by aggregated QoS policies in the network domain. This dynamic process repeats itself until reaching a stable state.

TABLE I
Definition of the Parameters [7] [8] [12]

Name	Notation
Service (or Application)	$\{S_1,S_2,S_3,..,S_n\}$
Shared Resources	$\{r_1,r_2,r_3,..,r_m\}$
Maximum Resources	$\{R_1,R_2,R_3,..,R_m\}$
QoS Requirements	$\{Q_{i1},Q_{i2},Q_{i3},..,Q_{nm}\}$
QoS Achieved	$\{q_{i1},q_{i2},q_{i3},..,q_{nm}\}$

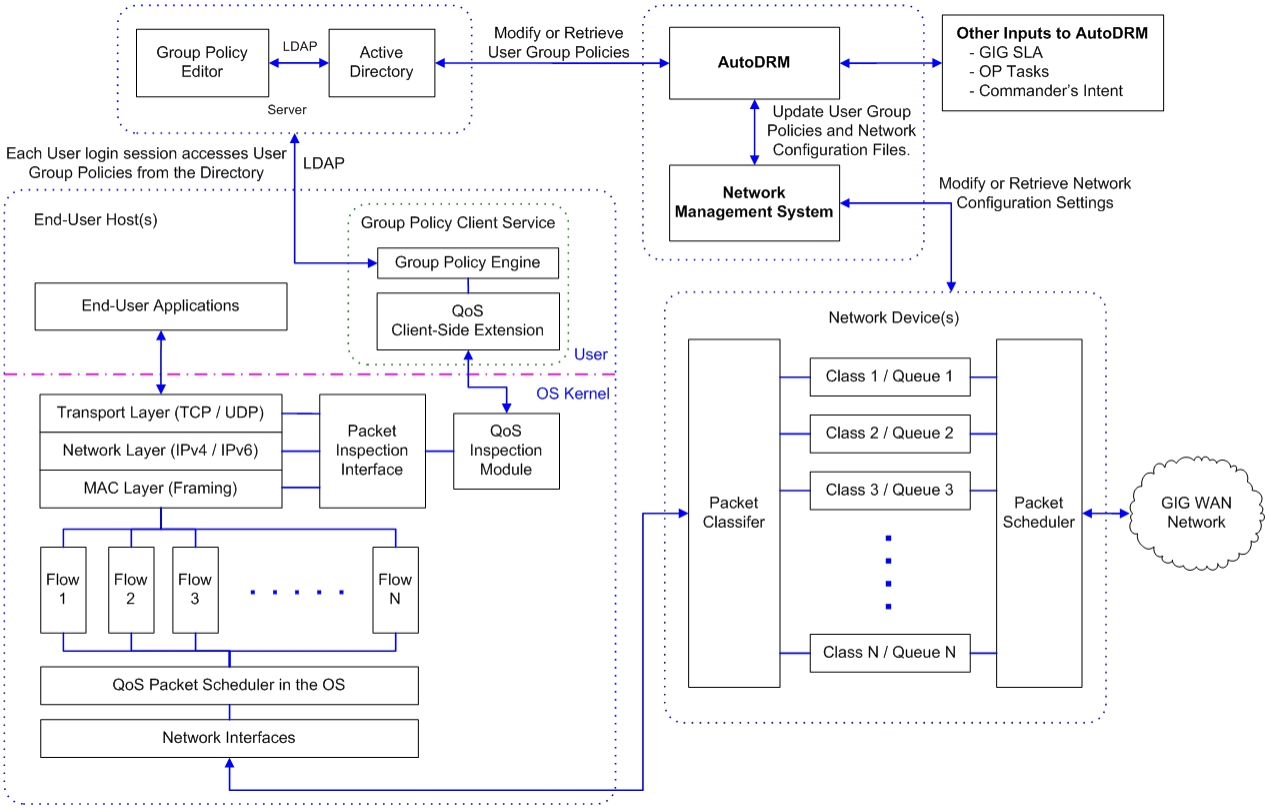


Fig. 3. AutoDRM QoS Architectural Concept

III. Network Architecture

A. Tactical Edge Network

The current *TSCE* system implements the Navy's open architecture strategy and has achieved a full commercial off-the-shelf (COTS) solution. Inspired by the concept of Service Oriented Architecture (SOA), the upcoming *CANES* system aims to perform more tactical functions while reducing the physical footprint by consolidating the network architecture across multiple security enclaves [5]. *TSCE* and *CANES* are distributed real-time enterprise systems typically deployed on tactical afloat units. Both systems provide various services for the user communities in the tactical edge networks. Since these services are deployed in a common network, they often compete for shared resources. Examples of these shared resources in the systems include processor units, memory devices, storage disks, networking devices, security devices, and others. To ensure that each service is performed at adequte QoS levels, the AutoDRM function is required in these edge networks.

B. End-To-End Network Prototype Test Bed

An experimental ETE network prototype test bed as shown in Fig. 2 was developed in order to host the AutoDRM system. According to the operational view in Fig. 1, the prototype network consists of four representative network domains: Shipboard Edge network, GIG Wide Area Network (WAN), GIG-N networks, and GIG-N edge network. The shipboard edge network is simulated by a Local Area Network (LAN) group consisting of several PCs, a network switch, and a gateway router. For the purpose of simplifying the prototype test bed development, the router between the shipboard edge network and the GIG WAN is shared. Two Virtual LANs (VLANs) were configured to represent the respective network domains. In practice, a gateway router from the shipboard edge network is connected to another router residing in the GIG WAN domain. A satellite communication (SATCOM) simulator configured with the same settings as in [13], is used to simulate the network characteristics across a long latency SATCOM link. An OPNET System-In-The-Loop (SITL) [14] scenario was developed to simluate the latency and packet loss rate in the GIG-N networks. The GIG-N edge network is assumed to mirror the shipboard edge network in the prototype test bed.

IV. AutoDRM Software Architecture

AutoDRM interfaces with a NMS to provide dynamic resource management capability for the shipboard edge networks. Fig. 3 illustrates the QoS concept in the AutoDRM architecture. In the context of QoS, network devices such as routers and switches can be conceptually represented with a network packet classifier, various queues, and a packet scheduler. Depending on the priority tag marked in each packet header and number of traffic classes, incoming packets are examined and classified into different outgoing queues. Departures of the outgoing packets are scheduled using a queuing technique (*e.g.* weighted fair queuing). QoS policies using

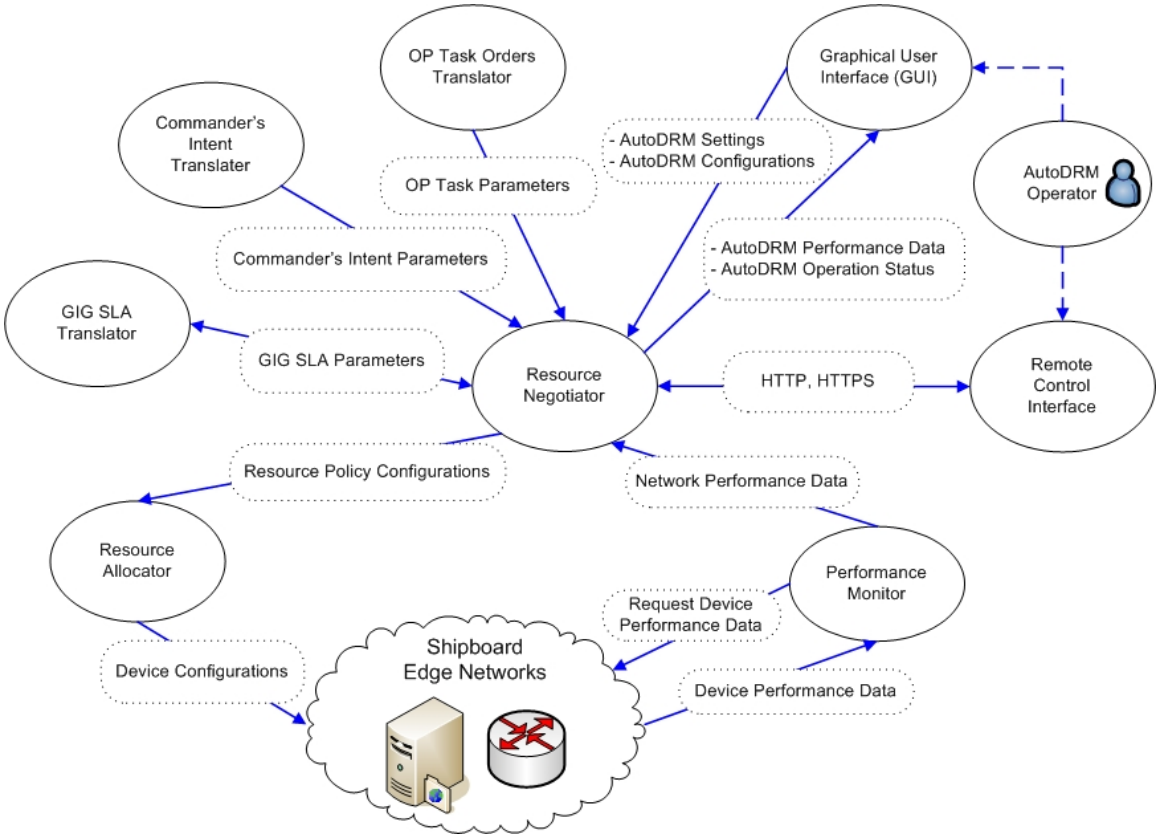


Fig. 4. AutoDRM Functional Block Diagram

differentiation service (DiffServ) [15] [16] can be configured in each network device to control the behavior of these QoS related components.

As shown in Fig. 3, an end-user host system (*i.e.* PC, server..etc.) executes heterogeneous user applications while utilizing services provided by a common set of network protocol stacks (*i.e.* TCP/IP) in the operating system kernel [17]. AutoDRM can retrieve and modify user group policy objects (GPO) which are stored in the Active Directory of the network domain server. Each GPO defines the QoS parameters such as data throttle rate and Differentiated Services Code Point (DSCP) value at the application level on a per-user or per-computer basis. Upon authentication of a user login session, the group policy client service retrieves user group policies from the Active Directory server. Depending on the user's privilege or the IP address of a host system, GPO determines the behavior of the network traffic generation from each application. More technical details of this policy-based QoS architecture is described in [17]. Based on the dynamics of the network performance measures, AutoDRM utilizes the technology by remotely updating the GPOs via scripts.

To accomplish the QoS objectives in AutoDRM, Fig. 4 illustrates functional components of AutoDRM which includes Graphical User Interface (GUI), remote interface, several input translators, resource negotiator, performance monitor, and resource allocator. The standalone GUI provides an interface for the system administrator and mission planning operator to perform initial setup and any intermediate system-level update. The remote user interface provides a convenient method to remotely control the AutoDRM software. The following subsections describe the functions of the translators, resource negotiator, performance monitor, and resource allocator.

A. Input Translator

Several built-in translators are required for AutoDRM in order to parse and translate structured documents containing Commander's Intent [18], operational task orders, and GIG Service Level Agreement (SLA) [19]. Commander's Intent describes the network and application performance parameters required by the shipboard commander on the afloat platform. The operational task orders are mission specific and are the derivatives of communication plans for tactical units. The input parameters from GIG SLA can be derived from Service Level Specifications (SLS) which is a subset of GIG SLA. The SLS specifies the communication parameters for the edge user community to subscribe to the GIG networks. The format of these inputs are assumed to be in structured documents (*e.g.* XML). Fig. 5 illustrates an example of the Commander's Intent schema. The operational task orders and GIG SLA are assumed to be formatted in a similar fashion. A parsing function first parses through the structured documents to extract a set of key communication attributes. A translating function then converts the extracted attributes into measurable parameters and stores the information in a translator database.

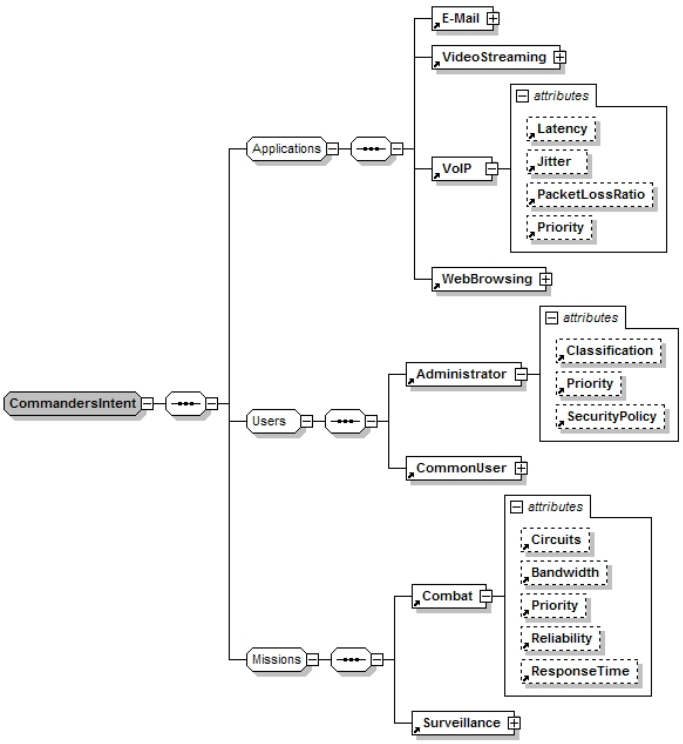


Fig. 5. An Example of Commander's Intent Schema

B. Resource Negotiator

The core function of AutoDRM is the resource negotiator, which determines the resource allocation based on real-time network performance measurements and notifications from NMS. The resource negotiator exploits a real-time near-optimal heuristic algorithm to determine the resource assignments. Fig. 6 depicts the functional block diagram of the resource negotiator. There are five fundamental functions in the resource negotiator: data fetch function, parameters mapping, sorting based on prioritization, resource assignments, and results transformation.

The data fetch function retrieves translated parameters such as bandwidth, packet delay, packet loss and other measurable network parameters from the translator database and the performance monitor database. The mapping function maps input parameters into a multidimensional array data structure where each array represents a services set, a resources set, a QoS set and other required data sets. The sorting function uses sorting algorithms (*e.g.* quick sort, binary sort..*etc.*) to efficiently sort each data set based on the prioritization of each item within the data set. The real-time near-optimal heuristic algorithm takes the mapped data set and gradually assigns resources to each task in order to minimize the error between the requested QoS levels and the actual QoS levels [9]. The results from the algorithm will be transformed in the results transformation function. The main objective of the results transformation function is to convert the acceptable resource assignments into actual user group policies and/or network configuration formats that can be used to regulate the QoS behavior of the devices in the edge network domain. A centralized database

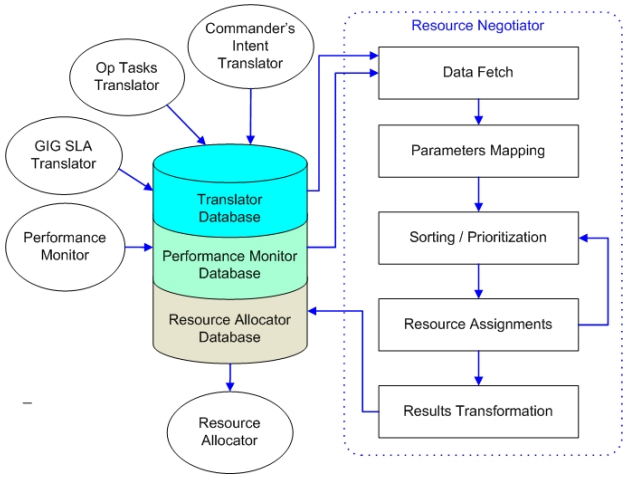


Fig. 6. AutoDRM Resource Negotiator Functional Block Diagram

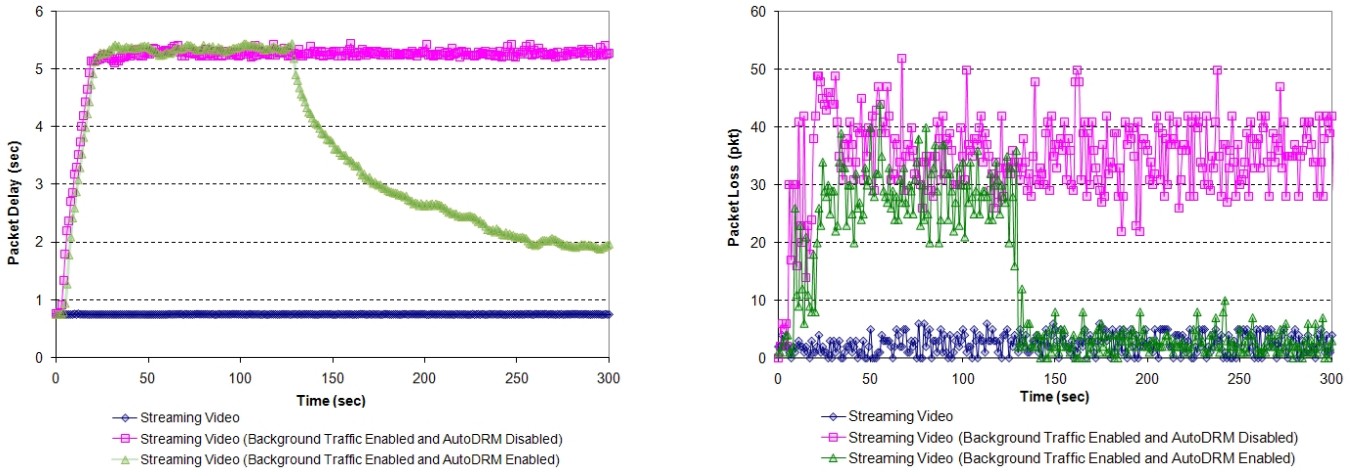
is shared among translators, a performance monitor and the resource allocator. A scheduler in the resource allocator will then update the GPOs in the network domain via a scripting method and/or network device configuration changes via NMS.

C. Performance Monitor

The performance monitor in AutoDRM exploits the rich set of network monitoring features in NMS to collect real-time network performance measurements through Simple Network Management Protocol (SNMP). The major categories of these features include service polling, data collection, events and notifications [20]. By leveraging the monitoring features in NMS, the performance monitor function constantly retrieves network performance information from the NMS into the performance monitor database in AutoDRM. In addition, the performance monitor also interfaces with the Active Directory server containing the GPOs. Modification of any GPO is updated in the database as well. Database updates are monitored by the resource negotiator. In the event of any network performance change (*e.g.* degraded bandwidth, increase of packet loss..*etc.*), the resource negotiator re-evaluates the QoS requirements and re-computes resource assignments to mitigate the possibility of resource contention in the network domain. The performance monitor makes use of common remote scripting tools and the development toolkits provided by the NMS for developing the required software interfaces.

D. Resource Allocator

Similar to the performance monitor, the resource allocator in AutoDRM interfaces with the NMS to modify the network configurations in each individual network device as well as updating user group policies in the network domains by remote scripting methods. In the event of network performance changes, the resource negotiator responds with an updated resource allocation and stores the necessary changes into the shared resource allocator database. The resource negotiator sends out notifications to the resource allocator which then updates the configurations of affected network devices and



(a) Packet Delay (sec)

(b) Packet Loss (pkt)

Fig. 7. Experimental Results

the host systems. The resource allocator is primarily responsible for scheduling the configuration updates in the network domain.

V. Experimental Setup

A use case consisting of three test scenarios was developed and experimented in the prototype test bed in order to investigate the effectiveness of the AutoDRM system. The use case assumes a tactical community user in the shipboard edge network is receiving a mission critical streaming video service from a server residing in the GIG-N edge network. The network traffic flow representating a streaming video service is simulated as an User Datagram Protocol (UDP) flow using Distributed Internet Traffic Generator (DITG) [21]. A NMS deployed in the shipboard edge network monitors real-time network performance. The NMS is pre-configured to genereate asynchronous notifications (*i.e.* SNMP traps) to the AutoDRM system when certain network thresholds are met (*i.e.* packet delay $\geq$ 5 seconds, packet loss $\geq$ 20 pkts). Upon receiving the notifications, AutoDRM determines that the streaming video service has the highest priority among other background traffic flows. Thus, the system provides preferential service to the streaming video flow by allocating more resources to it. The AutoDRM resource allocator achives this goal by updating the policy-based QoS parameters in the Active Directory as well as updating QoS policies in the configurations of the router and the switch.

For the purpose of performance comparison, three test scenarios were performed. Each test scenario has different network traffic conditions. The first test scenario consists of a single streaming video traffic flow without loading any background network traffic. This test scenario established a baseline test result. Two test scenarios consisting of a streaming video traffic flow with background traffic flows were also performed. One test scenario was with AutoDRM enabled and another test scenario was with AutoDRM disabled. The background network traffic contains nine heterogeneous UDP and TCP flows. Since the performance of the mission critical streaming video was under investigation, the packet delay and packet loss are the two key QoS performance metrics of interests in this experiment.

VI. Results and Discussion

The experimental results for each test scenario are shown in Fig. 7. The measurements are primarily focused on the QoS performance of the streaming video traffic flow over an initial period of five minutes. The QoS performance results include the packet delay as shown in Fig. 7(a) and the packet loss as shown in Fig. 7(b). As illustrated in Fig. 7(a), the baseline end-to-end packet delay for the streaming video flow without any background network traffic is 0.75 seconds. When the streaming video service is running with background network traffic and with AutoDRM system disabled, the end-to-end packet delay is quickly surged to more than 5 seconds after 20 seconds of run time. The packet delay stays at 5 seconds throughout the remaining duration of this test scenario. With AutoDRM system enabled, the NMS generates asynchronous notifications when the packet delay exceeds 5 seconds at 20 seconds of run time. The packet delay for the streaming video flow starts to decrease after 130 seconds of run time. The packet delay becomes stable at about 2 seconds after 250 seconds of run time.

As shown in Fig. 7(b), baseline packet loss for the streaming video flow is less than 8 packets at any time interval. With AutoDRM system disabled, streaming video flow running with background network traffic results in packet loss ranges from 20 to 50 packets after about 10 seconds of run time. With AutoDRM system enabled, the NMS generates asynchronous notifications when the packet loss exceeds 20 packets at 10 seconds of run time. The streaming video flow running with background network traffic shows improvement of packet loss to less than 10 packets at 130 seconds of run time. It can be concluded that AutoDRM system takes more than 100 seconds to improve the both QoS performance metrics of the streaming video flow in this experimental setup.

VII. Conclusion

An architecture of Automatic Dynamic Resource Management system has been developed in this study. An end-to-end network prototype test bed is also developed to host the AutoDRM system. A use case consists of three test scenarios with different network traffic conditions was experimented in the prototype test bed. The test results demonstrate improved QoS performance for a high priority streaming video service by deploying AutoDRM system in a tacical edge network. The results show that AutoDRM system is a required function to dynamically improve the network QoS performance. The system will improve the networking efficiency of the future tactical edge network. It will also enhance the capability of tactical shipboard network to be one step closer towards an ETE GIG architectural framework.

Our interests are focused on developing an approach for providing ETE QoS management services through automatic and dynamic mechanisms. To Achieve this, it is important to recognize the necessity to incorporate network performance and connectivity data with service request information associated with the application and the specific services available or supported by the network. In the future, we propose to use the developed prototype test bed for exploring any protocol mechanism that support this objective specifically recognizing the need to related network performance to the Commander's Intent.

Acknowledgment

The authors are grateful to numerous colleagues at Lockheed Martin, MS2 and University of Minnesota for reviewing the technical content and providing feedback on this paper. They also thank anonymous reviewers for providing comments to improve the overall quality of this paper.

Disclaimer

The opinions, recommendations, results, conclusions, and findings in this paper do not necessarily reflect the official views and positions of Lockheed Martin Corporation.

References

[1] A. Kantawala, D. Voce, and D. Gokhale, "QoS Architecture for Session Oriented GIG Applications," in *Proc. IEEE Aerospace Conference*, Big Sky, Montana, Jul.4–11 2006, p. 9.

[2] C. Gedo, Y. Xue, J. Evans, C. Dee, and C. Christou, "GIG QoS Inter-Domain Interoperability Challenges," in *IEEE Military Communication Conference*, Orlando, FL, Oct.29–31 2007, pp. 1–7.

[3] (2006, Aug.) Global Information Grid Net-Centric Implementation Document: Quality of Service (T300).

[4] M. Albuquerque, A. Ayyagari, M. A. Dorsett, and M. S. Foster, "Global Information Grid (GIG) Edge Network Interface Architecture," in *IEEE Military Communication Conference*, Orlando, FL, Oct.29–31 2007, pp. 1–7.

[5] A. S. Peng, B. R. Eickhoff, T. He, and D. J. Lilja, "Toward Consolidated Tactical Network Architecture: A Modeling and Simulation Study," in *IEEE Military Communication Conference*, San Diego, CA, Nov.16–19 2008, pp. 1–7.

[6] P. Lardieri, J. Balasubramanian, D. Schmidt, G. Thaker, A. Gokhale, and T. Damiano, "A Multi-layered Resource Management Framework for Dynamic Resource Management in Enterprise DRE Systems," *Journal of System and Software*, Jul. 2006.

[7] R. Rajkumar, C. Lee, J. P. Lehoczky, and D. P. Siewiorek, "A Resource Allocation Model for QoS Management," in *IEEE Real-Time Systems Symposium*, Dec.2–5 1997, pp. 298–307.

[8] ——, "Practical Solutions for QoS-based Resource Allocation Problems," in *IEEE Real-Time Systems Symposium*, vol. 2074, Dec.2–4 1998, pp. 296–306.

[9] F. Harada, T. Ushio, and Y. Nakamoto, "Adaptive Resource Allocation Control for Fair QoS Management," *IEEE Trans. Comput.*, vol. 56, no. 3, pp. 344–357, Mar. 2007.

[10] B. Dasarathy, S. Gadgil, R. Vaidyanathan, A. Neidhardt, K. P. B. Coan, A. McIntosh, and F. Porter, "Adaptive network QoS in layer-3/layer-2 networks as a middleware service for mission-critical applications," *Journal of System and Software*, Sep. 2006.

[11] M. M. Akbar, E. G. Manning, G. C. Shoja, and S. Khan, "Heuristic Solutions for the Multiple-Choice Multi-Dimension Knapsack Problem," in *Proc. Int. Conf. Computational Science*, May 2001, pp. 659–668.

[12] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A Scalable Solution to the Multi-Resource QoS Problem," Carnegie Mellon University, PA, Tech. Rep. CMU-CS-99-144, May 1999.

[13] A. S. Peng and D. J. Lilja, "Performance Evaluation of Navy's Tactical Network using OPNET," in *IEEE Military Communication Conference*, Washington, DC, Oct.23–25 2006, pp. 1–7.

[14] "OPNET Model User Guide," Version 14.5, OPNET, 2008.

[15] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475, Dec. 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[16] J. Babiarz, K. Chan, and F. Baker, "Configuration Guidelines for DiffServ Service Classes," RFC 4594, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4594.txt

[17] Microsoft. (2006) Policy-based QoS Architecture in Windows Server "Longhorn" and Windows Vista. [Online]. Available: http://www.microsoft.com/technet/community/columns/cableguy/cg0306.mspx

[18] L. G. Shattuck, "Communicating Intent and Imparting Presence," in *Military Review*, Mar. 2000, pp. 66–72.

[19] B. Doshi, P. Kim, B. Liebowitz, K. I. Park, and S. Wang, "Service Level Agreements and QoS Delivery in Mission Oriented Networks," *White Paper, MITRE Corporation*, May 2006.

[20] OpenNMS. (2009) About the OpenNMS Project. [Online]. Available: http://www.opennms.org/index.php/FAQ-About

[21] "D-ITG V.2.6.1d Manual," Version 2.6.1d, University of Naples Federico II, 2008. [Online]. Available: http://www.grid.unina.it/software/ITG/codice/D-ITG2.6.1d-manual.pdf