A PRELIMINARY IMPLEMENTATION OF A CONTENT–AWARE NETWORK NODE

N. Vorniotakis, G. Xilouris, G. Gardikis,
N. Zotos, A. Kourtis

National Center for Scientic Research "Demokritos"
Institute of Informatics and Telecommunications
Patriarchou Gregoriou st, Ag. Paraskevi
Athens 165 10, Greece
nkvorn@iit.demokritos.gr

E. Pallis

Department of Applied Informatics and Multimedia
Technological Educational Institute of Crete
Estauromenos, 715 00 Heraklion, Crete, Greece
epallis@pasiphae.teiher.gr

ABSTRACT

This paper presents a preliminary implementation of a content–aware network node as part of a Content–Aware capable network infrastructure. The proposed network node facilitates all the possible functions provided by an ordinary router or gateway while at the same time exploits flow– and content– awareness in order to identify the content carried within a flow and handle it efficiently. In this context this paper discusses the main concepts and operating principles around the flow and content awareness and presents a preliminary implementation, build around the Linux operating system.

Index Terms— content–aware, MANE, QoS, media networks, flow-aware, DPI

1. INTRODUCTION

Nowadays the trend in network management is to push intelligence to the edges of the network infrastructure, while at the core continue to provide fast switching and forwarding supporting the incremental needs for speed and availability. The constant evolution in hardware capabilities — in terms of CPU power and memory availability — is an enabling factor in order to design network equipment capable of baring the effort of identifying the content transferred inside the packet flows and applying specific policies or routing respectively.

In the same time multimedia content, is anticipated to be increased by at least a factor of 6, rising to more than 990 Exabytes before 2012, fueled mainly by the users themselves [1]. In order for future network architectures to cope with this versatile content and services, future network equipment should become content-aware. This capability will offer more functionalities to the network management entities in order to efficiently manage the network and offer flexible and dynamic traffic handling and QoS assurance functions.

This work presents a preliminary implementation of a content–aware network node based on Linux operating system. This content–aware network node is seen as an enhanced Media-aware Network Element (MANE). As defined in [2], MANE is a network element, such as a middle–box or application layer gateway that is capable of parsing certain aspects of the RTP payload headers or the RTP payload and reacting to the contents. One of the most prominent uses of MANE is along with H.264/SVC encoded flows for in–network adaptation of RTP flows in an effort to enable efficient network usage and dynamic adaptation to available network capacity. The implemented network node exploits deep packet inspection (DPI) algorithms in order to provide the necessary content-awareness functionalities. Depending on the content type, the node control module enables i) the policing and differentiation of the incoming traffic flows and ii) routing table selection and forwarding.

Finally a preliminary implementation of the proposed content-aware network node is tested in laboratory setup.

The paper is structured as follows: Section 2 discusses the content–awareness enabling functions, Section 3 presents the design and implementation of the content-aware network node, Section 4 presents the testbed and the validation scenarios and finally Section 5 presents the results.

2. CONTENT–AWARE ENABLERS

2.1. Flow awareness

Content classification could be performed per flow of network data. This can reflect in much smaller processing delays since, if the content of a flow is classified, then every subsequent packet of that flow arriving at the router can be instantly forwarded to the appropriate destination. No further processing is needed. So only the minimum amount of packets of a flow, in order to determine the service, need to be processed, resulting to reduced processing delay.

The flow management can be applied only on Ingress routers using a system with a hash table. This will remove the need for keeping track of flows on all nodes of the core network and result on a more scalable system.

2.2. Traffic Classification

Traffic classification techniques, in current telecommunication networks, span from the exploitation of Layer 3 information to Layer 7. Beyond this, many techniques combine multilayer information with application data inspection (DPI) in order to provide accurate traffic identification [3]. Another approach less invasive but with complex implementation, is the exploitation of methods that do not relay on the packet protocol stack information but also on statistical flow information [4]. Currently, there is not a definite solution that can address all the cases, triggering much research in these fields [5][6][7].

A survey of the methods to classify traffic at an application level includes [8]:

- Exact matching, i.e., infer the application identity by assuming that most applications consistently user well-known TCP and UDP port numbers (e.g., 80 for HTTP, 22 for SSH etc.).
- Prefix match: the rule field could be a prefix of the header field; this could be used also to get statistics about packets originating from a sub-network
- Heuristic methods, using Deep Packet Inspection (DPI) [5], i.e., looking for application-specific data (or well-known protocol strings) within the TCP or UDP payloads
- Machine learning based on statistical features:
  - Supervised Learning, i.e., Naive Bayes, Decision Tree, Nearest Neighbors (NN), Linear Discriminate Analysis (LDA), Quadratic Discriminate Analysis (QDA), Bayesian Neural network
  - Unsupervised Learning, i.e., EM, AutoClass and K-Means.

2.2.1. DPI

The DPI approach suggests that network flows are inspected and information is extracted from higher layers of packet data (up to application layer) [9]. By exploiting the rich high layer information that is provided through DPI, content–aware network node control functions will provide various options in packet flow handling in respect to policing, shaping, QoS etc.

The exploitation of DPI methods for traffic classification is build around two basic assumptions: Third parties unaffiliated with either source or recipient are able to inspect each IP packets payload (i.e., is the payload visible?) The classifier knows the syntax of each applications packet payloads (i.e., can the payload be interpreted?)

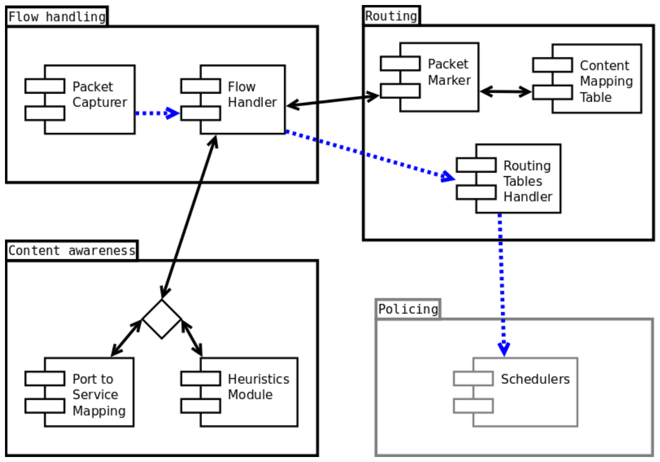


Fig. 1. Application modules.

A library of protocol signatures and filter strings has to be available to facilitate the content identification. This library may be consulted, so the protocol can be accurately detected. Depending on how this library is build, the detection can be arbitrarily accurate. However, the larger it gets the more resource hungry the classification procedure grows. The signature library needs to be constantly updated as application protocols evolve or as new protocols emerge. Relying on an out-dated library will have a severe impact on the accuracy of the identification.

3. DESIGN AND IMPLEMENTATION

Content–awareness system functions were designed to be modular and scalable, so it will be able to cope with the high performance demands needed for the proposed functionality.

The modules of the application and their interconnection are illustrated in Figure 1. The incoming packets are following the path denoted with the dotted arrows to reach the next routing node. The other modules manage the flows in parallel and try to perform a mapping of the content of each flow to a service class.

The Flow Handling module comprises of the Packet Capturer module, that captures incoming network packets, and the Flow Handler that organizes incoming packets to network flows.

The Routing module comprises of the Packet Marker and the Routing Tables Handler. The Packet Marker marks the flows that are to be policed — according to the configuration send by the Network Management — so the kernel routing system may apply specialized forwarding. The Routing Table Handler is the module that can create new routing tables and policies so that the subsequent packets of the flow are policed as needed.

The Content Awareness Module depends on the Heuristics functions to determine the service of each flow using

Deep Packet Inspection (DPI) techniques or Port to Service Mapping for simple services identification.

Finally the Policer Module is in charge of applying the desired policies at the respective flows. This module includes also the queue schedulers that are used for traffic control (shaping, differentiation, prioritization).

3.1. Flow Handling Module

The Flow Handling module is responsible for detecting the network flow each incoming packet belongs to so that a policy can be applied per flow rather than per packet. This approach removes the need to process each packet individually; if a packet belongs to an already classified flow it is not processed resulting to big savings in terms of processing delay.

Network flows at a point in time can be uniquely identified by the 5–tuple of the sender address and port, the destination address and port and the transport layer protocol used. This 5–tuple is used to in order to calculate a hash key in order to keep the reference of this flow to a Flow Table that keeps track of the incoming flows detected. A flow idleness timer is incorporated in order to expunge the idle flows from the Flow Table, hence saving memory resources.

3.2. Content Awareness Module

The Heuristic module is the basis of the content–awareness functionality. It monitors the state of a network flow and uses heuristics to determine the content of the flow. Following the content identification for a given flow a Content Mapping Table is consulted, that indicates the policies and prioritization to be applied.

A simple version of the RTP dissector is illustrated on Figure 2 using a flowchart. As the incoming packets are traversing the processing loop, each one is checked to determine whether it belongs to an already established flow. If it belongs to a new flow it goes through some tests to determine whether it is an RTP packet. The version illustrated is a two pass control. First, the packet needs to be a UDP packet on an even port. If this is true, the SSDP and the packet timestamp are stored and the flow created for the packet is marked so it will have to go through the second check. When another packet arrives for the flow and reaches the second control branch, its SSDP is checked to be equal with the previous and the timestamp difference is 1. If those hold then the flow is marked as an RTP stream.

This is a simplified version of RTP detection that works quite well but is only serves as an example. The actual detection algorithm is much more complex so it can be accurate on most cases since it has more passes and also takes into account RTCP data.

If the heuristic method can not classify the flow or if it is not desired to use DPI, the Port to Service mapping module can be used that uses the IANA ports–to–service suggestions.

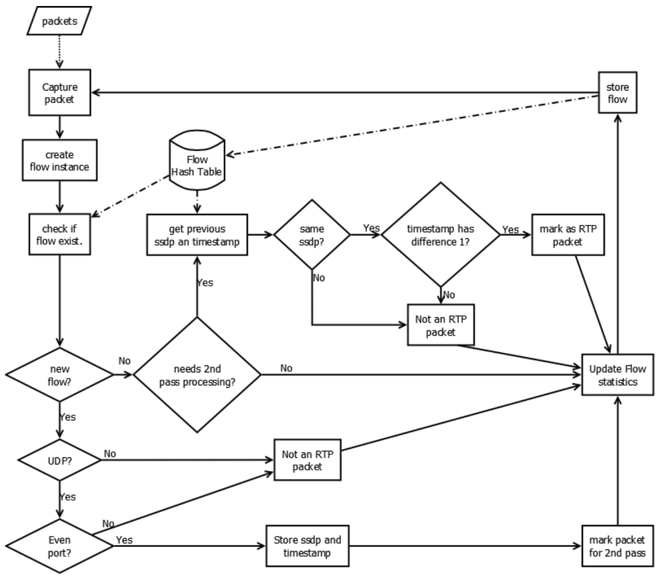


Fig. 2. RTP dissector.

3.3. Routing Module

This module is responsible for applying at the incoming, identified flows the appropriate internal marking in order for other modules to apply the appropriate policies and proceed to indicated actions. In general for every incoming flow, after the content is identified three main decisions need to be made. First being the how to police the traffic at the ingress interface, secondly how to route the flow and last how to handle (shaping, conditioning, prioritization) the flow at the egress. This module exploits functionalities provided by the Linux OS kernel and User Space utilities (i.e. iptables, traffic control).

The Network Management entities that control the all these aspects through a Content Mapping Table that contains information on how to police and condition the flows depending on content type.

In order to achieve Content Based Routing a number of alternative local RIBs is used that are statically pre-assigned. The CMT contains the mapping of each content type to the appropriate RIB. The flow is marked appropriately depending on the current policy for the content and is forced to be forwarded using the routing tables created for the type of service it is categorized as. In this preliminary implementation static RIB were used, however a dynamic routing protocol may be used for the update of the routing tables so the system can adapt to the requirements of the network as they change.

4. TESTBED

The architecture of the test–bed used is illustrated on Figure 3 , the part of the network that is on the left of the ingress router is the access network of the server which relies on the

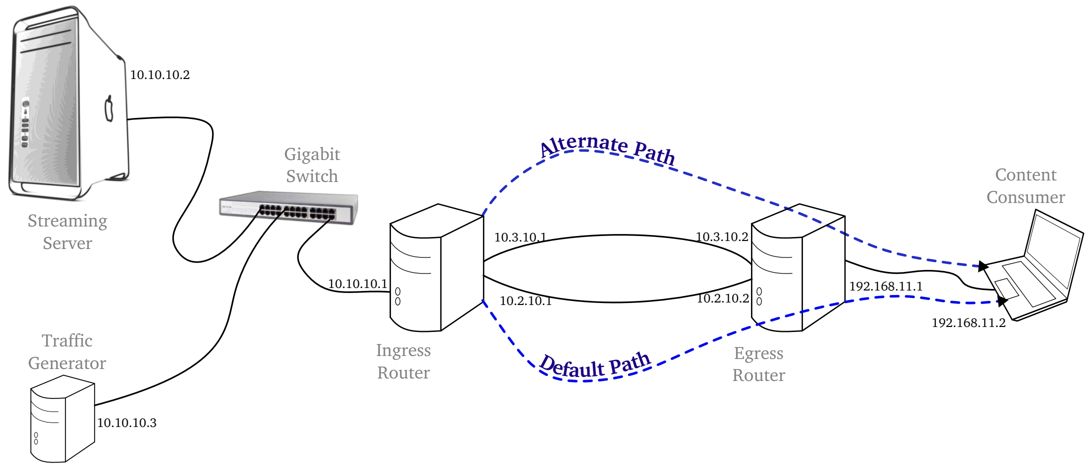


Fig. 3. Topology and configuration.

domain 10.10.10.0/24. The part on the right of the egress router is the access network of the client which is the domain 192.168.11.0/24. For the sake of simplicity, this network has only a single link from the egress point to the client.

The core network is composed of the ingress router, the egress router and the two links between them. The default path is the Gigabit link on the domain 10.2.10.0/24. The alternative path is a 100Mbps link on the domain 10.3.10.0/24.

This setup was used to emulate real world scenarios where services that ought to have special treatment on the core network, are being severely degraded in terms of quality by the losses in cases of heavy traffic. The sets of measurements performed try to show whether the content–aware functionality achieved to be beneficial for the prioritized services and if yes at what level.

The general concept is that the streaming server streams a flow of a video using RTP to the client that is considered to be a high priority service. The traffic generator is used to create a gradually increasing source of background traffic up to the point that the network reaches its limits and there are losses on the ingress interface. In the case the network is unable to distinguish the high priority service (i.e., no content–awareness) so the service will not receive any special treatment (i.e., forwarded via the alternative uncongested path). On the other hand, when the contentaware software is running, the high priority service can be detected. Then a rule is created and the subsequent packets of the priority flow will be forwarded via the uncongested interface.

4.1. Testing Scenarios

On an attempt to prove the usefulness and performance of this approach, two testing scenarios were employed. Each scenarios are divided in three cases; making use of pfifo schedules with no classification, using an HTB queueing discipline with SFQ schedulers on the leaves and re–routing specific class of services via different paths.

The first Scenario tests the behavior of the throughput of the network as it gets congested and how it is improved using traffic classification and re–routing traffic. The second measures the end–to–end delay and losses of the tree cases.

Another set of measurements was performed so that the end–to–end delay and losses can be surveyed. A media stream was transmitted to the client using RTP. The packets were dumped on the client and server and their timestamps were compared so that the delay could be calculated. Moreover, the sequence number of the RTP packet header was used to detect lost packets.

This sort of measurement required accurate and synchronized clocks so the timing difference of the packets is correctly calculates. Therefore both the client and server ware performing synchronization requests to NTP time servers so their clocks would have the minimum difference possible.

4.1.1. The default case

In this case a classic content–agnostic network is emulated. All traffic is routed via the default path including important services. This case is used as a reference of comparison to the next scenarios that employ content–aware functionality.

4.1.2. The HTB case

This set–up examines the case where a privileged service is detected using the content–aware modules and is policed using a higher priority class. The HTB scheduler is used to provide class full forwarding.

In this case a RTP flow is used. Classic port–based content classification techniques would fail to classify the service because services were detected using destination port numbers. However, RTP cannot use a standard port number so a heuristics based classification must be used.

Once the flow of the priority service is detected, it is marked so the Linux kernel forwarding plane can use the policy defined for this particular content for the subsequent packets of the flow.

4.1.3. The content–aware routing case

In this case the high priority service is forwarded to a different path upon detection. This path could be a better behaving link for the type of service that is forwarded. For example it could have more bandwidth available lower delay or jitter etc.

For this case, different routing tables are created for each class of service. When the network flow of a managed service is detected, it is marked so it is forced to be forwarded using the routing table for the service it belongs.

5. RESULTS

5.1. The default case

In this scenario the pfifo queue was employed to the ingress router. The background traffic was again gradually increased until the ingress link was congested.

The results are illustrated on Figure 4.a. The aggregate traffic is rate limited to the ingress interface at 100 Mbps. Since this limit is surpassed, the FIFO queue starts dropping packets randomly. The lost packets are represented on the figure with green impulses.

The congestion results to a very big queueing delay which after a threshold starts increasing linearly until it gets very high a almost 1200 ms. This very high delay, combined with the packet loss, makes this link unusable for services that are sensitive to these factors like interactive applications, VOIP or media streaming. What is worse, there is no special treatment to important services, so mission critical packets could be lost.

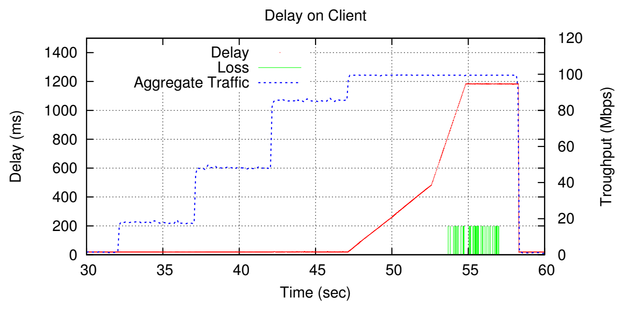
5.2. The HTB case

In this case the HTB structure was used again with SFQ schedulers. This allows for prioritized services so the video streamed will be marked as having a higher priority so it will not be affected of the backgroud traffic.

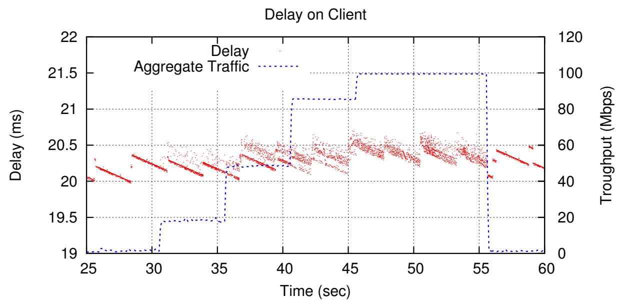
The results are illustrated on Figure 4.b. The aggregate traffic is limited to 100 Mbps but this time there is no lost packets for the privileged service. Moreover, there is not any remarkable increase on the end–to–end delay of the stream. There is only an slightly increased scattering on the delay values indicating a slight increase on the jitter value.

5.3. The content–aware routing case

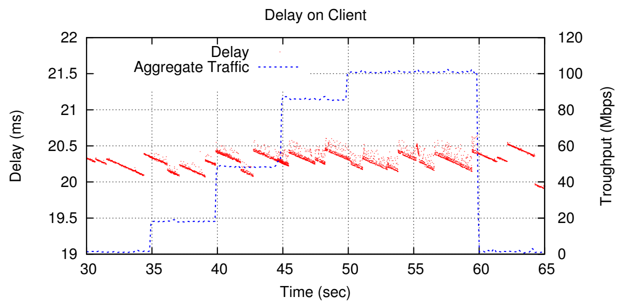
In this case, the content aware routing functionality is used to forward the media stream trough the alternate path.



(a) Delay of video stream (pfifo)



(b) Delay of video stream (HTB with SFQ)



(c) Delay of video stream (routing)

Fig. 4. Results of the three scenarios.

The results are illustrated on Figure 4.c. In this case only the background traffic is following the default path and is rate limited. The privileged service is following the alternate path and therefore has not lost any packets.

The delay increase is again negligible. The scattering of the delay values is also smaller indicating less jitter.

6. CONCLUSIONS

This paper presented a preliminary implementation of a content-aware network node. Moreover a test–bed was used in order to validate the implementation either in the case where the identified content is differentiated and prioritized from the background traffic or in the case that the identified flow is routed from another physical interface. Further work is going on to fully implement a Linux based content-aware node in the framework of the FP7 research project ALICANTE.

7. ACKNOWLEDGEMENTS

This work was supported by the EC in the context of the ALICANTE project (FP7-ICT-248652)

8. REFERENCES

[1] Media Delivery Platforms Cluster, "Multimedia delivery in the future internet: A converged network perspective," 2008.

[2] S. Wenger, MM Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RFC 3984; RTP payload format for H. 264 video," *IETF, February*, 2005.

[3] M. Zhang, W. John, K. Claffy, and N. Brownlee, "State of the art in traffic classification: A research review," in *PAM Student Workshop*, 2009.

[4] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *Neural Networks, IEEE Transactions on*, vol. 18, no. 1, pp. 223–239, 2007.

[5] A. Callado, C. Kamienski, G. Szabo, B. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on internet traffic identification," *Communications Surveys & Tutorials, IEEE*, vol. 11, no. 3, pp. 37–52, August 2009.

[6] J. Garca-Nieto, J. Toutouh, and E. Alba, "Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 5, pp. 795 – 805, 2010, Advances in metaheuristics for hard optimization: new trends and case studies.

[7] Stenio F. L. Fernandes, Carlos Alberto Kamienski, Judith Kelner, Dłnio Mariz, and Djamel Sadok, "A stratified traffic sampling methodology for seeing the big picture.," *Computer Networks*, vol. 52, no. 14, pp. 2677–2689, 2008.

[8] T.T.T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, 2009.

[9] Klaus Mochalski and Hendrik Schulze, "Deep packet inspection," White paper, dpacket.org, `http://www.ipoque.com/userfiles/file/DPI-Whitepaper.pdf`, 2009, [Online; accessed 12-February-2010].

[10] ICT-ALICANTE, "Media ecosystem deployment through ubiquitous content-aware network environments," No248652, `http://www.ict-alicante.eu`, 2010.