Quality of Service (QoS) Protocols over IPv6 Networks: Rewriting OPNET Modules to Implement RSVP in IPv6

Stephen Frost
Michael Gawlas

CS756-001
Fall 2006
Prof. Pullen
Quality of Service (QoS) Protocols over IPv6 Networks:
Rewriting OPNET Modules to Implement RSVP in IPv6

Purpose of Project

The original project proposal was to obtain Quality of Service (QoS) models/levels for IPv6 on a simulated network to best mimic what might be exhibited over a WAN/MAN environment. This objective was to be reached through the use of the simulation program -- OPNET.

This objective was never fully achieved due to several hurdles that were discovered within the OPNET modeler. The IPv6 features of their product did not allow or have the implementation of RSVP over IPv6. The results we do have for this project is the rewritten OPNET source code which implements RSVP support for IPv6.

Application Background

OPNET Modeler is commercial-level network modeling and simulation environment, supporting control over all layers of the network model from the physical connection error rates to the application settings. Also, there are a large number of support and model packages. It is a very robust product, allowing the user to edit entire networks and nodes within the network, while viewing and editing state models of a process and the code to run those states.

The version of the package that we used for our development was OPNET Modeler version 12.0.A (Academic License). We also had a license for the IPv6 module.

Previous Work in this Area

Dr. Pullen, George Mason University, has previously conducted extensive work in prior years in the field of IP Quality of Service simulations. These previous simulations did not involve IPv6. Details on these previous projects can be found at: https://netlab.gmu.edu/qosip.htm

Previous student's of Dr. Pullen attempted to conduct models and simulation within IPv6 in OPNET. These models and code were made available for our project. These models did not work and there were several issues with them. The project was built on OPNET version 11.5 and included some modified code for the Resource ReSerVation Protocol (RSVP) protocol. When this project was run, no RSVP traffic was generated, and several warnings and errors were present in the system. Some of these warnings involved dropped or erroneous packets, improper RSVP bandwidth allocations, and private link addresses trying to be routed on public/backbone networks.

Discovery of the Extent of Previous Issues

Our initial work involved taking the previous student's project models and converting them to be used on OPNET v12.0 with the IPv6 module. Our first area of concentration was to troubleshoot the errors and warnings that were reported in the simulation logs. There were over 100 errors and warnings. Many of the errors were reduced by enabling the RSVP protocol and IPv6 addressing on all of the network nodes interfaces and on all network routers. Surprisingly,
these settings were not turned on for all of the nodes, but this was due to the fact that there were several places where it needed to be turned on within each node and within certain global variable and attribute settings.

During our analysis, we discovered that private (link-local) IPv6 addresses, beginning with "FE80::", were trying to be routed on backbone and public internet routers. We made sure the proper routing protocols were enabled on all the routers and this error was resolved. We traced through several simulations at this point and discovered that some RSVP packets were being generated per the protocol. These initial messages were the path messages sent by the requester. Unfortunately, these packets were addresses with IPv4 addresses that could not be routed on an IPv6 network. After ensuring all IPv4 addressing and routing was disabled on our model, our doubts about the OPNET Modeler became a reality—it does not natively support RSVP in IPv6.

Additionally, while the prior work on modifying the RSVP source code to support IPv6 was significant, we determined that it did not appear to be a good base to start from given the changes in OPNET v12.0 and the apparent intent that the RSVP module only supported IPv6. Thus, the stock RSVP module in OPNET v12.0 was used as the base code while the previously modified RSVP source code was used as a reference.

Simulated Network Overview

The network model we used was a modified version of the previous students test/debug model. It is a small test WAN consisting of one backbone, three ISP domains, and six LAN domains (Figure 1). Each LAN was fairly unique, but each had a gateway router, a hub, and two to three hosts/workstations (Figure 2).

![Global Network Model](image)

**Figure 1 - Global Network Model**

The backbone network (Area 4) consisted of two routers connected via an OC-12 connection. The ISP domains (Area 1, 2, & 3) were connected to the backbone networks via DS-3 connections. The individual LAN domains were connected to the area border routers via DS-1 connections. These details were several changes to the original model, such all connections were DS-1
connections and the backbone (Area 4) network did not exist. [BBR1 and BBR2 were connected via ABR3, such that the actual backbone network was really BBR1, ABR3, and BBR2, and also directly served LAN’s F5 and F6].

Figure 2 - LAN Network Model

As stated, each LAN consisted of an Ethernet forwarding gateway (router), an Ethernet hub, and at least two workstations. All LAN interconnections were via 100BaseT cable. Some domains contained specifically identified FTP servers or workstations configured to host or support specific application-level programs. Our model included: File Transfer/FTP (light and heavy loaded), Video Conferencing (low and high quality bandwidth settings), and Voice over IP (low and high quality settings). For several of the workstations, these applications were configured to conduct the service with reserved resources and bandwidth implemented using the RSVP protocol.

Modifying the RSVP Protocol and OPNET Source Code

The base OPNET v12.0 RSVP code was not written to support IPv6. We suspected that a valid IPv6 address was being passed to the protocol, but OPNET was casting the IPv6 address into an IPv4 address. Modifying the source code to support IPv6 required the recoding and modifying the code and data structures within several pieces of OPNET. A pictorial view of RSVP state model is exhibited in Figure 3.
Figure 3 - RSVP Protocol State Model

Modified RSVP-related source code files include:
- rsvp.h
- rsvp.pr.c
- rsvp_log_support.h
- rsvp_log_support.ex.c
- rsvp_mem_support.h
- rsvp_mem_support.ex.c
- rsvp_msg_support.h
- rsvp_msg_support.ex.c
- rsvp_state_support.h
- rsvp_state_support.ex.c
- rsvp_support.h
- rsvp_support.ex.c
- rsvp_app_support.h

External modules which required modification include:
- ip_output_iface.pr.c
- ip_rte_support.ex.c
- tpal_intf_rsvp.pr.c
- tpal_intf_tcp_v3.pr.c
- tpal_intf_udp_v3.pr.c

Each of these external files and modules were modified to support both IPv4 and IPv6 addressing. The above code consisted of over 45,000 lines of code, approximately 1000 of which had to be changed or added. After completing the modifications to the RSVP source code the other native OPNET modules needed to be modified to support IPv6 for RSVP. These modules, related to multicasting, transport-application interface and IP-to-Interface translations, assumed RSVP supported only IPv4. They were recoded to support both IPv4 and IPv6 for the RSVP module. The modified code for all of these modules will be included in an appendix to this report. Additionally, the OPNET model files will also be included.

Current Status
Once the individual code modules had been modified to have matching or compatible data structures for the IP addressing, the protocol code and associated IP address modules were recompiled and relinked. The simulation will now run the entire "10 minute" scenario. RSVP PATH messages (IPv6 formatted) are initiated and sent across the network to their destination. These messages are picked up at each of the routers which then set up the appropriate RSVP state information to support the path. RSVP Resv messages were then sent back in reply again using IPv6 through the routers which updated their RSVP state data. Lastly, RSVP Conf messages were sent back using IPv6. The RSVP system then logged successful RSVP path completion messages in the DES log.

Unfortunately, there was not enough time to reconfigure the simulation to force RSVP block states or overload the circuits enough to show dropped packets against non-RSVP enabled sessions. Future work in this area is now possible under IPv6 and it is hoped that future users of OPNET with RSVP will benefit from this work. The DES log from the simulation run showing the successful RSVP path completion messages and the simulation console log showing an ltrace of the RSVP module will also be included as appendixes.

Knowledge Gained

Though we did not complete the intended objective, we did gain a better understanding of IPv6, the Quality of Service/RSPV protocol, how to use and modify the advertised aspects of OPNET, and the intimate knowledge of the source code supporting OPNET. Under IPv6, we learned about the specification in general, but specifically about the addressing scheme that makes it unique from IPv4. More importantly, knowing which address ranges are specified for private or link-local addressing and for multi-casting was important in differentiating the services and packets on the network. Also, IPv6 has a different discovery method during the network initialization phase.

With regards to the RSVP protocol, we learned about this in depth. One interesting item about how it works is that it does not actually route the traffic, but it simply determines the path (essentially a virtual circuit) \textit{[path message from the requester to the server]} and reserves bandwidth along that circuit \textit{[resv message from the server to the client]}. Additionally, the protocol regularly sends out follow on messages to preserve or modify the virtual circuit path or bandwidth.

Lastly, this project has demonstrated the power and capabilities that OPNET holds as a network modeler and simulator. Our assumption that the RSVP module in OPNET v12.0 contained support for IPv6 protocol was not true. While the more common features of OPNET work great, adding features has a very sharp learning curve and requires extensive coding, often outside of the specific module where the feature is being added.

Closing

This project wasn't as successful as we wished, but it did offer valuable insights into IPv6, QoS schemes, and network simulation. We also offer the source code and our programming notes to future students to continue this project to fruition.