RSVP-SQOS : A SECURE RSVP PROTOCOL

Vanish Talwar *and *Klara Nahrstedt*

Department of Computer Science
University of Illinois at Urbana Champaign
{vtalwar,klara}@cs.uiuc.edu

Suman Kumar Nath

Department of Computer Science
Carnegie Mellon University
sknath+@cs.cmu.edu

ABSTRACT

Distributed multimedia applications require end-to-end quality of service(QoS) in order to be accepted and used. One approach to achieve end-to-end QoS is to provide end-to-end resource reservations. Resource ReSerVation Protocol (RSVP) [3] [1] is a unicast and multicast signalling protocol for setting up network bandwidth reservation. In this paper, we propose a solution for securing RSVP messages in a flexible, efficient and scalable manner. Our solution extends the RSVP protocol with a scalable QoS protection, using a hybrid hierarchical security approach. The RSVP messages go through two different protocol treatments - one within subnetworks and the other across subnetworks. We use delayed integrity checking within the subnetwork by sending feedback messages from the egress node. A stronger integrity and encryption check is made on messages sent across subnetworks. Our solution is thus an intermediate approach between the extremes of hop by hop authentication [2] and the SDS/CD protocols [5],and overcomes the drawbacks of the two protocols.

1. INTRODUCTION

Distributed multimedia applications require end-to-end quality of service(QoS) in order to be accepted and used. One approach to achieve end-to-end QoS is to provide end-to-end resource reservations. Resource ReSerVation Protocol (RSVP) [3] [1] is a unicast and multicast signalling protocol for setting up network bandwidth reservation. RSVP sets up a distributed temporary state in routers and hosts and is used by a host to request specific QoS from the network for particular application data streams or flows. It is also used by routers to deliver QoS guarantees to all nodes along the path(s) of the flows and to establish and maintain state for the requested service. The RSVP protocol raises several security issues [3] [2] [5] :
(1) Message Integrity and node(router) authentication (2) User Authentication , (3) Non-Repudiation , (4) Confidentiality , (5) Replay Attacks , (6) Traffic Analysis (7) Cut and Paste Attacks, and (8) Denial of Service Attacks

Current solutions for securing RSVP are not complete in themselves and have their own pros and cons. [2] has the drawback of high overhead, expensive key management and vulnability against malicious router attack. [5] has the drawback of bad scalability. We propose a solution, RSVP with Scalable QoS protection (RSVP-SQoS) protocol, using a hybrid approach applied to a hierarchical network. A simplified first version of our solution was presented in [6]. The current paper presents an enhanced version of the RSVP-SQoS, concentrating on flexibility and scalability of the protocol. Further, the protocol detects intrusions as quickly as possible with a low overhead. The analysis results show that the protocol detects an intrusion within time, proportional to twice the size of the largest subnetwork in the path. The number of extra messages generated for every Path or RESV message is atmost equal to the number of subnetworks in the path. We describe our RSVP-SQoS protocol in the next section.

*This research was supported by the NSF EIA 99-72884, Airforce grant under contract number F30602-97-2-0121.

2. RSVP WITH SCALABLE QOS PROTECTION (RSVP-SQOS)

2.1. Protocol Design

The entire network is composed of a number of subnetworks (or domains), labeled S_i ($1 \leq i \leq N$), such that *border nodes* (routers) of a subnetwork are well defined. The border node which a message uses for entering into (or quiting from) the subnetwork,is called *ingress* (or *egress*, respectively) node of the subnetwork for that particular message. The nodes other than the border nodes are *core nodes*. There can be multiple ingress/egress nodes in a subnetwork and the same node may act as an ingress node for some messages in one direction and egress node for some other messages in other direction. The RSVP-SQoS protocol achieves a secure RSVP transmission of QoS parameters in three phases. Within each subnetwork S_i, we run Phase 1 of RSVP-SQoS for the PATH messages and Phase 3 of RSVP-SQoS for RESV messages. Across the egress node of a subnetwork and the ingress node of the next subnetwork, we run Phase 2 of RSVP-SQoS. The secure PATH message protocol, running along the path from data's source to data's destination, is using Phase 1 and Phase 2 of RSVP-SQoS, and is shown in Figure 1. The secure RESV message protocol, running along the path from data's destination to data's source, uses Phase 2 and Phase 3, and is shown in Figure 2. Throughout the protocol, it is assumed that whenever a router detects some intrusion, an alarm is raised to a local policy decision point in the router. This local policy decision point will decide whether the RSVP message should be dropped or not. Also it will decide whether it is appropriate to distribute a TearDown message. Let I_k and E_k refer to the ingress and egress nodes of the k'th subnetwork, used by the message under consideration. I_k and E_k may be different for the PATH and RESV messages, depending on the direction of the message under consideration. PATH_FB$_k$ and RESV_FB$_k$ refer to the feedback messages (corresponding to a PATH and RESV messages) in this k'th subnetwork. Subscripts are assumed to be in increasing order from the sender to the receiver for all the messages (PATH, RESV and feedbacks). Below

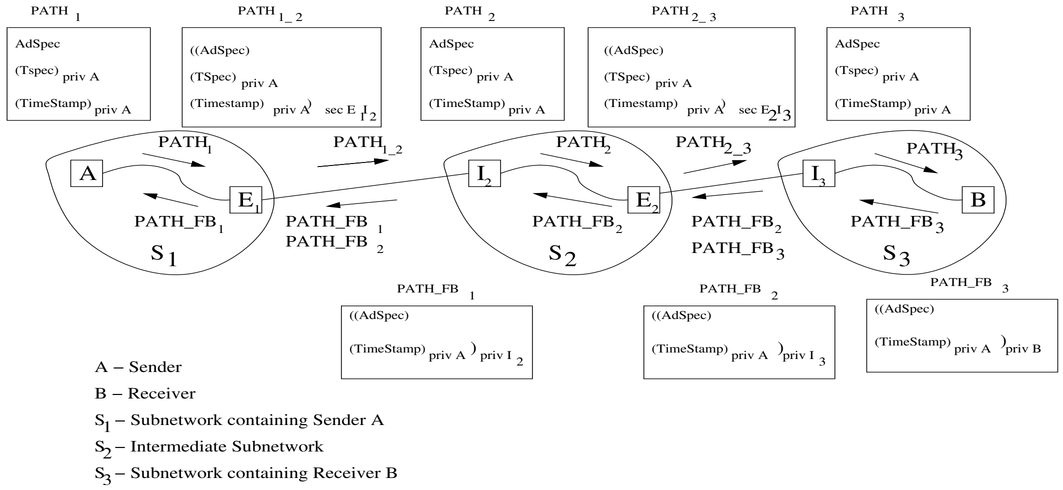

Figure 1: The PATH messages using Phase 1 and Phase 2

are the three different phases for the k'th subnetwork.

Phase 1 (Intra Domain PATH protocol)
(1) (Sender or Ingress Node) If $Sender_{Alice}$ is in the subnetwork, it digitally signs the non-mutable parameters of the PATH message, puts the timestamps and sends the message towards the E_k. If the current subnetwork is an intermediate subnetwork, I_k performs three tasks. **(a)** On receiving the PATH message from the E_{k-1} (using Phase 2 of the protocol), it forwards the message towards E_k. **(b)** On receiving the PATH message from the E_{k-1}, it creates the feedback message PATH_FB$_{k-1}$ by digitally signing the `Adsepc(PATH)` it just has received from E_{k-1} using Phase 2 of the protocol and sends it back to E_{k-1}. **(c)** On receiving PATH_FB$_k$ from downstream, it forwards the message towards E_{k-1}.
(2) (All nodes) On receiving the PATH message, each node (border or core nodes) of the subnetwork receives PATH message from the upstream, authenticate the sender and verifies the timestamp before processing the PATH message. The authentication is done by verifying the Alice's signature using her public key. The verification of the timestamp is used to detect replay attacks. On success, it passes the message ahead, otherwise it raises an alarm to its local policy decision point. On receiving the PATH_FB$_k$ message from downstream (signed by I_{k+1}) it verifies whether `Adspec(PATH)` is equivalent to or lower than the last `Adspec (PATH)` it forwarded downstream. If there is a conflict, it raises an alarm to its local policy decision point. Otherwise, it (if it is not $Sender_{Alice}$) forwards the feedback message upstream. If the node is the last node($Receiver_{Bob}$), the message is processed as outlined below.
(3) (Receiver or Egress Node) If the current subnetwork contains $Receiver_{Bob}$, on receiving the PATH message, $Receiver_{Bob}$ makes the feedback message PATH_FB$_k$, sends it back towards I_k and starts Phase 3. Otherwise, E_k performs following three tasks: **(a)** On receiving the PATH message, it forwards the message to I_{k+1}, using Phase 2. **(b)** On receiving the PATH_FB$_k$ from I_{k+1}, it verifies if `Adspec(PATH)` is exactly the same as what it forwarded last time, and on success, forwards it towards I_k. **(c)** On receiving PATH_FB$_{k+1}$ from I_{k+1}, it verifies whether the `Adspec(PATH)` is equivalent or lower than the `Adspec(PATH)` it forwarded downstream. In any case of conflict, it raises alarm to its local policy decision point.

Phase 2 (Inter Domain PATH/RESV protocol)
(1) The egress and ingress nodes of all subnetworks decide on a shared key between every (egress,ingress) node pair that are connected to each other. This could be decided based on a key management strategy and could be cached for optimization. The actual key management strategy is not specified in the protocol.
(2) A key hash of the entire PATH (or RESV) message is made using this key by the egress node. The particular hashing algorithm to be used is not specified here, but examples are MD5, HMAC-5 etc. Since the non-mutable components of the messages are digitally signed, a key-hash of only the mutable components may be required. The keyed hash is attached to the message(PATH or RESV). If required, the message (PATH or RESV) could also be encrypted for confidentiality. The message(PATH or RESV) is then sent to the ingress node.
(3) On receiving the message from the egress node of the previous subnetwork, the ingress node creates the keyed hash of the message using the shared key and verifies it with that attached to the message. If they do not match, an intrusion must have occurred and an alarm is raised to the local decision point.
(4) If the keyed hash matches, the ingress node authenticates the Sender and verifies the timestamp before processing the PATH (or RESV) message. The authentication is done by verifying the sender's signature using the sender's public key. The verification of timestamp is used to detect replay attacks. On success, it forwards the message ahead.
(5) As in [2], we could maintain a security association for each authenticating key shared between an ingress and egress node. In this case, the key would be the unique number for a given sending router and would be used to determine the security association to use.

Phase 3 (Intra Domain RESV protocol) (Here I_k and E_k corresponds to the direction of the RESV message and may be different from the I_k and E_k for the PATH message).
(1) (Receiver or Ingress Node) If $Receiver_{Bob}$ is in the subnetwork, it digitally signs the RSpec parameters of the RESV message, puts the timestamps and sends the message towards the E_k.

If the current subnetwork is an intermediate subnetwork, I_k performs three tasks. **(a)** On receiving the RESV message from the E_{k+1} (using Phase 2 of the protocol), it forwards the message towards E_k. **(b)** On receiving the RESV message from the E_{k+1}, it creates the feedback message RESV_FB$_{k+1}$ by digitally signing the `RSpec(RESV)` it just has received from E_{k+1} using Phase 2 of the protocol and sends it back to E_{k+1}. **(c)** On receiving RESV_FB$_k$ from downstream, it forwards the message towards E_{k+1}.

(2) (All nodes) On receiving the RESV message, each node (ingress, egress or intermediate nodes) of the subnetwork receives RESV message from the upstream, authenticates the sender and verifies the timestamp before processing the PATH message. The authentication is done by verifying the Bob's signature using his public key. The verification of the timestamp is used to detect replay attacks. On success, it passes the message ahead, otherwise it raises an alarm to its local PDP. On receiving the RESV_FB$_k$ message from downstream (signed by I_{k-1}) it verifies whether `RSpec(RESV)` is equivalent to or lower than the last `RSpec (RESV)` it forwarded downstream. If there is a conflict, it raises an alarm to its local PDP. Otherwise, it (if it is not $Sender_{Alice}$) forwards the feedback message upstream.

(3) (Sender or Egress Node) If the current subnetwork contains $Sender_{Alice}$, on receiving the RESV message $Sender_{Alice}$ makes the RESV_FB$_k$, forwards it towards I_k and starts Phase 1. Otherwise, E_k performs following three tasks: **(a)** On receiving the RESV message, it forwards the message to I_{k-1}, using Phase 2. **(b)** On receiving the RESV_FB$_k$ from I_{k-1}, it verifies if `RSpec (RESV)` is exactly the same as what it forwarded last time, and on success, forwards it towards I_k. **(c)** On receiving RESV_FB$_{k-1}$ from I_{k-1}, it verifies whether the `RSpec(RESV)` is equivalent or lower than the `RSpec(RESV)` it forwarded downstream. In any case of conflict, it raises alarm to its local PDP.

2.2. Security Analysis

In this section, we will explore how the protocol ensures security, i.e. detects tampering or malicious injecting. A malicious router along the path can tamper some parameters of the RSVP objects in two ways: it can viciously decrease or increase the QoS value of the flows (e.g. `AdSpec(PATH)`). Here, we will consider only the malicious increase of the values. We do not consider the malicious decrease of some QoS value for the following two reasons. First: in RSVP, decreasing the value of some object (e.g. `Ad-Spec(PATH)`) by some router is perfectly legal – this router conceptually does not have much resources to offer. Second, in fact, if a RSVP-enabled router is compromised or faulty, we should avoid using its services/resources anyway. Decreasing the QoS value will actually discourage flows being established through this vicious or faulty router. A detailed description of different types of malicious attacks on RSVP messages can be found in [5].

It is obvious that if the constant part of a message is tampered by some router along the path, it will finally be detected by the receiver, since it is digitally signed by the sender. Now consider the mutable part of the message. Suppose some core router in subnetwork k tampers (i.e. increases the value of) a QoS parameter in the RSVP message, e.g. `AdSpec (PATH)`. When it reaches I_{k+1}, or the receiver, a signed feedback message will be sent through the inversed route path and every router (up to E_{k-1} or the receiver) on the path will have a chance (and optionally) to examine whether the history is consistent with their local observation. So if some core router maliciously increased the value of `AdSpec(PATH)`, and this tampering affects the final resource reservation, then this tampering would be detected by the router just before this malicious router along the path of the feedback message in the following manner.The feedback message containing the value of `AdSpec(PATH)` would be greater than the last value of `AdSpec(PATH)` the router had forwarded downstream[1]. Since the final value of the `AdSpec(PATH)` should decrease along the path, the router would detect the increase in the value of `AdSpec(PATH)` as illegal. Hence, tempering by some core router will always be detected.

Now let us consider the case where the ingress/egress routers are malicious. If the egress node E_k tampers some object, it will be detected when the feedback message reaches the router just before the egress node. Likewise, if the ingress node I_k tampers some object, it will be detected when the corresponding feedback message reaches E_{k-1}. Now consider a different tampering I_k (generator of the feedback message) node can perform: it can forward a message different from the message it sent as feedback. But this is not possible. Note that, to generate a feedback message, I_k is supposed only to sign the object it got from E_{k-1}, not to modify it. So, when the feedback message reaches E_{k-1}, any modification will be detected. Again, the message forwarded by the I_k node will eventually turned into a feedback message and will be forwarded back until it reaches E_{k-1}. So E_{k-1} can detect if I_k did some tampering during forwarding the message. So tampering by any ingress/egress node cannot go undetected.

Thus, the protocol can detect tempering by any of the routers on the path. Note that, the protocol also makes it difficult for a group of routers to be successfully malicious. For example, to make some undetected malicious tampering, all the routers in a subnetwork must be malicious and cooperative, otherwise the friendly router may detect the tampering. An ingress node might be successful in forwarding and feedbacking different version of a message only if the previous egress node cooperates with it, but since they are in different subnetworks, it is unlikely.

3. COMPLEXITY ANALYSIS

Let N be the number of subnets labeled S_i ($1 \leq i \leq N$) through which the PATH and RESV messages traverse for a given (Sender, Receiver) pair. Let $size_i$ ($1 \leq i \leq N$) be the number of hops traversed by the PATH and RESV messages within subnet S_i.

Message Complexity

The number of extra messages (feedback messages) that are generated for every PATH message (or RESV message without multicast flows) is atmost N. If there are M multicast flows, then the number of feedback messages for the RESV messages for all the multicast flows would be atmost M*N. But in practice, this value would be less if merges take place at intermediate subnetworks. The extra bits added in the PATH(or RESV) messages are those for integrity checking (digital signature of sender/receiver and integrity bits for subnet-to-subnet authentication). The feedback messages only carry the digitally signed AdSpec or RSpec values and are thus very small in size.

[1] If the value is smaller or equal, and the upstream routers have nothing to complain, then the increase by the malicious router must have been decreased by some other downstream routers. So, the malicious activity will not come into effect and can be ignored.

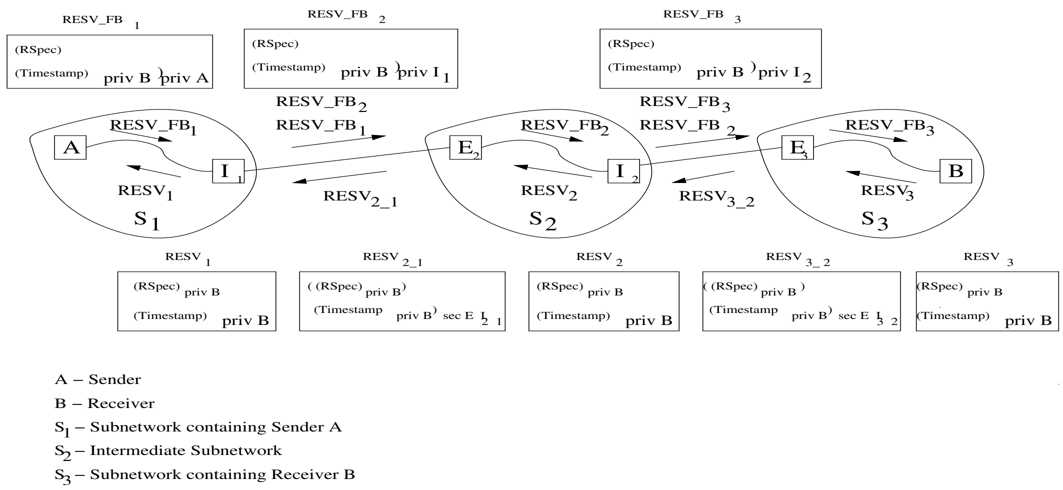

Figure 2: The RESV messages using Phase 3 and Phase 2

Time Complexity

Every new message created is digitally signed by the Sender/ Receiver. Further, every intermediate egress node also creates an integrity hash of each message. This is an overhead but is much lesser than the overhead of integrity hashing per hop per message in [2]. At every hop, the sender is authenticated and the timestamp is verified. The extra processing for the feedback messages at a node lies in the verification of AdSpec(or RSpec) values. The ingress node creates a feedback message containing the AdSpec(or RSpec) value digitally signed. The overall overhead in terms of time is more than [5] but less than [2]. However, note that our protocol is more scalable than [5]

If an intrusion occurs at node A within a subnetwork S and if A is n hops away from the egress node of subnetwork S, then the intrusion will be detected within time atmost equal to $2 * n *$ (Time to traverse one hop within a subnetwork) + $2*$(Time to traverse one hop across subnetworks). In the worst case, $n = max(size_i)$. If an intrusion occurs at the hop across subnetworks, then it will be detected at the ingress node of the next subnetwork.

There are three possible schemes to apply when an ingress node receives the PATH(or RESV) message. We can allow the ingress node to forward the PATH(or RESV) message while the feedback messages check for intrusion. Thus the intrusion detection takes place concurrently with the forwarding of the PATH(or RESV) messages and there is no time lost due to the feedback messages. However, if an intrusion is detected, unnecessary resource reservations would be made. The second scheme is to make the ingress node wait for the feedback message to finish intrusion detection before forwarding the PATH (or RESV) messages. Then the total extra time required would be atmost $2*$ max($size_i$) $*$ N $*$ (Time to traverse one hop within a subnetwork) + $2*$N $*$(Time to traverse one hop across subnetworks). In this case, no unnecessary reservations would be made. The third scheme would be to allow the ingress node to forward the PATH(or RESV) messages but give higher priority to the TearDown messages so that they could chase the PATH (or RESV) messages in case of intrusion. At the time the feedback message detects an intrusion and sends a TearDown message, the PATH(or RESV) messages would be atleast $2 * (n + 1)$ hops away. The TearDown messages would have to cover up this distance in as minimum time as possible. For example, if the TearDown messages are given higher priority so that it could be processed at twice the rate at which the PATH(or RESV) message is processed, then the TearDown message will meet the PATH(or RESV) message within a time of $\frac{2*(n+1)}{Rate of processing PATH(or RESV) message}$. This scheme tries to minimize unnecessary reservations while preventing any delay at the ingress nodes.

4. CONCLUSIONS

In this paper, we have proposed a solution for securing RSVP messages,that is flexible and scalable for larger networks. The RSVP-SQoS design views the network as a set of subnetworks with different security assumptions within a subnetwork and across subnetworks. The RSVP-SQoS protocol minimizes the delay in detecting intrusions with low overhead.

5. REFERENCES

[1] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation Protocol – V.1 Applicability Statement *RFC 2208*, Sep. 1997.

[2] F. Baker, B. Lindell, and M. Talwar. RSVP Cryptographic Authentication. *RFC 2747*, Jan. 2000.

[3] R.Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol – V.1 Functional Spec. *RFC 2205*, Sep. 1997.

[4] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. *RFC 2401*, Nov. 1998.

[5] Tsung-Li Wu, S. Wu, and F. Gong. Securing QoS: Threats to RSVP Messages and Their Countermeasures. *IEEE IWQoS*, pp. 62–64, 1999.

[6] Vanish Talwar and Klara Nahrstedt. Securing RSVP For Multimedia Applications *Proceedings of ACM Multimedia (Multimedia Security Workshop)*, Los Angeles, Nov. 2000.