Network Working Group   Keith McCloghrie
Internet Draft   Michael Fine
                                                            Cisco Systems
                                                          22 October 1999

A Comparison of Policy Provisioning Protocols

draft-kzm-policy-protcomp-00.txt

Status of this Memo

Distribution of this document is unlimited. Please send comments to the RAP Working Group at rap@iphighway.com.

Copyright Notice

1. Introduction

The IETF's RAP Working Group has almost completed its task of defining COPS as a standards-track protocol for RSVP Admission Policy. The WG's charter is now being extended to cover standardizing a policy provisioning protocol, and the proposal put forward for consideration by the WG is based on defining a new Client-type for COPS. This new Client-type is intended to work in conjunction with the QoS Policy objects and schema definitions being defined in two other IETF Working Groups: Differentiated Services, and Policy. Before proceeding any further, it has been requested that a comparison be undertaken as to why COPS is better suited to this than a (modified if necessary) version of SNMP. This memo attempts to document such a comparison.

2. Background Information on QoS Policy

Several years ago, the IETF took a step towards standardizing Quality of Service (QoS) by defining Integrated Services [INTSERV]. A part of that effort was to define RSVP [RSVP] as a standardized QoS signaling protocol. When RSVP is used, each router participating in the signaling can independently allocate local resources to an individual RSVP session, or reject a request when local resources are exhausted. However, local information is insufficient to make a decision to admit or reject an RSVP request based on network-wide policy. Such policy might be static, or alternatively, might depend on dynamic network-wide state information. With RSVP signaling, the policy does not need to be distributed to each and every router; rather, both the policy and any network-wide state upon which the policy might depend, can be kept at a (few) centralized Policy Server location(s); the receipt of RSVP signaling messages provides the opportunity for a particular router to ask its Policy Server for an admit/reject decision for that RSVP session. It also provides the opportunity for the Policy Server to respond with not only a decision, but also to specify other policy-driven actions (e.g., modifying or augmenting the policy information contained in the RSVP messages).

The use of COPS [COPS] as the protocol by which a router (the Policy Enforcement Point, or PEP) queries a central Policy Server (the Policy Decision Point, or PDP) on receipt of RSVP messages has been successfully tested in Interoperability Tests involving multiple vendors, and the specifications are currently under review by the IESG for progression to Proposed Internet Standard status.

A subsequent IETF effort towards standardizing QoS was to define Differentiated Services [DSARCH]. In its basic form, DiffServ specifies no signaling protocol. Rather, each packet gets marked with a DiffServ Codepoint (DSCP) and it is the DSCP which then determines the QoS treatment (the Per Hop Behaviour) which a packet receives. In this situation, a router does not have the opportunity to ask the centralized Policy Server when it receives the signaling message, since the packets arrive without being preceded by a signaling message. Therefore, the policy to be applied must be provisioned by the Policy Server. Note that such provisioning is not necessarily static, but may vary depending upon dynamic network-wide state (similar to RSVP policy).

The latest direction of IETF WGs includes the identification of scenarios and the definition of mechanisms whereby the two different QoS schemes (Integrated Services and Differentiated Services) can be used in tandem. One proposal involves how the path across the network of a particular session can include one or more domains of Integrated Services, as well as one or more domains of Differentiated Services [DSRSVP]. A related proposal involves using RSVP signaling as the means by which the specific DSCP value to be used in packets of that session can be communicated back towards the source [DCLASS]. Another proposes that both RSVP and DiffServ be used for the same sessions in the same domain: RSVP in the control plane for the application of admission control policy, and DiffServ in the data plane for specifying how a router treat the individual packets [DSRSVP]. A conclusion to be drawn from these proposals is that it is increasingly likely that there will be domains having a need for both RSVP-based policy as well as DiffServ-based policy, and that these two parallel types of policies will need to be consistent and coordinated, i.e., administered by the same set of Policy Servers.

Thus, in addition to being RSVP's policy protocol, COPS is also being proposed as the protocol for provisioning DiffServ policy [COPS-PROV], since using it for both facilitates the required consistency and coordination.

3. COPS

3.1. COPS Client-Types

COPS has a multiplexing mechanism whereby a single TCP connection can be used for multiple independent client-types. A client-type is defined for RSVP in [COPS-RSVP].

A different client-type is defined in [COPS-PROV] for provisioning. This client-type defines the concept of a Policy Information Base

(PIB) as the means by which to convey policy data. The first PIB [QOSPIB] is for a subset of QoS Policy which roughly equates to DiffServ.

3.2. The Structure of Policy Information and the PIB

Much like SNMP calls for a structure of management information and a Management Information Base of concrete management objects, COPS for provisioning calls for a structure of policy information (SoPI) and a Policy Information Base of concrete policy objects. SoPI and PIBs are intentionally like SMI and MIBs, in order to leverage knowledge of and experience with MIBs, but with a few intentional differences.

- PIBs are aimed at the definition of "higher-level" policy, e.g., network-wide policy, rather than the device-specific configuration and monitoring at which MIBs are aimed.

- PIBs are optimized for bulk configuration of multi-attribute objects; this is a big win.

3.3. Exclusive Access by the PDP

[COPS-PROV] specifies that one and only one PDP (Policy Server) controls a device (for a specific set of PIBs) at any one time. Giving one PDP exclusive access avoids SNMP's need to provide synchronization mechanisms to protect against multiple SNMP managers trying to access the same MIB objects at the same time. For example, this is the major reason why read-create tables in MIBs always have a RowStatus object. In contrast, PIBs for use with COPS do not need RowStatus to solve the multi-manager problem because only one PDP can be accessing the PIB at any one time.

Giving one PDP exclusive access also means that, during the time when COPS for Provisioning is enabled for a particular type of policy, SNMP and/or CLI are disabled from modifying any related configuration in the device. Of course, if configuration via SNMP/CLI becomes necessary, then COPS for Provisioning can be disabled (via SNMP or the CLI). Note that the PDP will recognize this because the COPS connection will be torn down.

An application, which configures a device using SNMP, can never be sure that the configuration it set several minutes/hours/days ago is still in effect, because it's possible that some other management application (or human) might have modified the configuration more recently. So, a prudent management application will periodically recheck that the configuration is unchanged. This need for re-checking is not specific to SNMP, but it is similarly required when configuration is done via CLI. In contrast, exclusive access by a single PDP avoids the uncertainity and consequent need for re-checking that the device's configuration has not been tampered with.

As well as not needing RowStatus, PIBs do not need the spin-locks (TestAndIncrement objects) which are included in some MIBs. This not only makes a PIB simpler, but it also reduces the complexity of the PDP's code as compared to a SNMP manager. For example, the following procedure is specified in [USEC] by which a manager creates a new (private and authenticated) user:

```
     1)  GET(usmUserSpinLock.0) and save in sValue.
     2)  SET(usmUserSpinLock.0=sValue,
             usmUserCloneFrom=templateUser,
             usmUserStatus=createAndWait)
         You should use a template user to clone from
         which has the proper auth/priv protocol defined.
     3)  generate the keyChange value based on the secret
         privKey of the clone-from user and the secret key
         to be used for the new user. Let us call this pkcValue.
     4)  GET(usmUserSpinLock.0) and save in sValue.
     5)  SET(usmUserSpinLock.0=sValue,
             usmUserPrivKeyChange=pkcValue
             usmUserPublic=randomValue1)
     6)  GET(usmUserPulic) and check it has randomValue1.
         If not, repeat steps 4-6.
     8)  generate the keyChange value based on the secret
         authKey of the clone-from user and the secret key
         to be used for the new user. Let us call this akcValue.
     9)  GET(usmUserSpinLock.0) and save in sValue.
     10) SET(usmUserSpinLock.0=sValue,
             usmUserAuthKeyChange=akcValue
             usmUserPublic=randomValue2)
     11) GET(usmUserPulic) and check it has randomValue2.
         If not, repeat steps 9-11.
     13) SET(usmUserStatus=active)
```

Notice that without the need to use usmUserSpinLock, the procedure would have 5 fewer steps, would have no loops, and three less values to be set. In fact, COPS would recommend the whole procedure be done in a single DEC message using a single OID and a single vector of values.

3.4. Different Design Philosophy compared to SNMP

Many of the SNMP protocol's design choices were based on the need for its use in debugging network problems (see section 6 of [RFC1270]). In contrast, the design choices for the COPS protocol have always assumed that the network is up and working. Thus, SNMP runs over UDP, whereas COPS runs over TCP. This choice of UDP versus TCP is one example of the difference in design philosophies.

3.5. COPS Takes Advantage of TCP

The use of TCP allows COPS to enjoy the use of large, and therefore more efficient messages (e.g., up to 64KB for each COPS object) spanning many TCP segments, without it needing to be aware of Path MTU. In contrast, SNMP-over-UDP requires an implementation to support a minimum of 484 bytes as its largest message size. Many SNMP agents support larger maximum message sizes than 484, but messages must still fit into one IP packet, in order to avoid the degradation in reliability of IP fragmentation. Thus, the use of SNMP messages larger than 1500 bytes is rare.

Like SNMP, each COPS message is a "transaction", meaning that all policies contained in one message must be successfully installed, or none of them are. However, COPS's much larger messages allow much more policy to be created, deleted and/or modified in one transaction, than would be possible with SNMP. With SNMP's limited message size, only small updates can be made atomically which could result in windows of time where the installed policy is inconsistent with itself.

Another consequence of SNMP's small message size is the potential that a complete row of a MIB table might not fit into one message. For example, a row containing six objects, each having a 255-byte long value, will overflow a 1500 byte message; and it takes only two such objects to overflow a 484 byte message. COPS's use of larger messages ensures that a whole row will fit into a single message. This allows COPS to create new rows/modify existing rows atomically by including the whole row at once, and in fact COPS only allows access to whole rows, not to the individual columnar objects within them. In contrast, SNMP must allow the option of creating/modifying rows by "dribbling" the individual values into the agent, one or more per message.

This potential for dribbled creation of an SNMP read-create table can result in many corner/error cases which have to be catered for in a

MIB implementation. The number of such corner/error cases is significantly reduced by the COPS requirement of atomic access to a row. Thus, support for a PIB table takes less code, and is easier and quicker to implement and test.

COPS also gains another advantage from requiring atomic access to whole rows. Since a whole row must be included in a COPS message, there is no need to include the name of each and every columnar object instance being access; instead, only the name of the row needs to be specified. So, COPS messages include an OID naming the row, plus a vector of values, one for each value in the row. In contrast, SNMP messages contain a varbindlist, consisting of the name and value of each referenced columnar object in the row. This results in a significant difference in message size, since OIDs are notoriously long. For example, consider a row consisting of 10 integers, with OIDs of length 15 bytes, and integer values of length 2 bytes: this would use up 210 bytes of an SNMP message, but only 57 bytes in a COPS message.

3.6. SNMP over TCP

Proposals to run SNMP over TCP have been put forward several times in the past, and each time have sparked debate. None of these past proposals were accepted enough to be published as RFCs. However, the result of one such debate is documented within [RFC1270].

The latest proposal is contained in [SNMPTCP]. It provides the interesting combination of using SNMP over UDP and TCP at the same time, with the sender of any initiating (i.e., not a Response) SNMP message choosing which transport protocol to use. Either side is allowed to tear down connections at any time if their resources are scarce, and non-initiators may refuse TCP connections whenever they wish. [As an aside: two statements in [SNMPTCP] would seem to be questionable: a) it's not clear why UDP necessarily increases latency due to small packet-sizes; and b) as RFC 1270 states, (some form of) retransmissions at the application level are needed even over TCP.]

Note, however, this latest proposal can not take full advantage of the use of TCP (e.g., the atomic access to rows described above), because the MIBs must still be able to be used with SNMP-over-UDP. Nor can it make use of only one OID for a whole row (and the resultant savings in message sizes), since it uses the same SNMP message format over both TCP and UDP.

3.7. COPS over TCP (Revisited)

By way of contrast, the use of TCP is fundamental to the way COPS for Provisioning works. COPS's Keep-alive operation code is used to ensure that at least one message is exchanged between the PEP and PDP within a timer value (specified by the PDP). (Note that these keep-alives serve as COPS's form of retransmission at the application-layer, see quote from RFC 1270 above.) If no message is received within the timer value, the connection is assumed to have failed and the PEP initiates a TCP connection to a Secondary PDP. If that connection attempt also fails (and while the PEP continues attempting to connect to both the primary and secondary), all policy which was provisioned during the previous COPS session is subject to an expiry timeout (set as part of the policy). That is, when the expiry timeout occurs before a COPS connection is re-established, the current policies expire with it. Thus, the policy itself is volatile and dependent upon the current state of the COPS TCP connection. For some policies, it is useful to have an infinite expiration time (which never expires); other policies may be ephemeral.

Note well that this expiry timer has no effect while the COPS connection is established. Obviously, while the COPS connection is established, the PDP can delete policies as and when they should expire. It is only when the COPS connection is lost such that the PDP cannot delete expiring policies, that use of the expiry timer is needed.

Again, SNMP is different: SNMP itself does not specify whether the creation or modification of values is volatile or non-volatile; instead, it leaves such specification to be part of the definition of MIB objects. One reason for this is that SNMP sets are used for both the setting of parameter values (which might normally be non-volatile), as well as a way of initiating actions (which of course are volatile).

3.8. Modifying Message Formats

SNMP has a very stable message format, which is not easily changed. Specifically, changes in this message format are not warranted for small gains in functionality, particularly if such changes were to alter the design parameters of the protocol. In fact, the last change in SNMP message format was to replace the uniquely-formatted SNMPv1 trap message with the SNMPv2_trap message having a format identical to all the other SNMP messages; this change was made specifically to make all the message formats the same (i.e., the only difference in ASN.1

format definitions is that the Response PDU uses two integers for error status and index, whereas the GetBulk PDU uses the same two integers for non-repeaters and max-repetitions).

In contrast, the COPS message format has only recently become stable, and some aspects of the message format are specific to individual client-types. Further, different COPS message have different sets of objects in them, and all values in a COPS message (except those in the message header) are formatted using a TLV format, such that new types can be specified in the future. That is, it is only the COPS message header which is common to all COPS messages.

Thus, COPS for Provisioning is at a point in time where changes in message format can be made for small but useful gains. An example of this is the inclusion in [COPS-PROV] messages of multiple error-codes, potentially one per individual row. Indeed, the values of these error-codes can be specific to the particular row's definition. That is, they can be specified as part of the definitions in a PIB, with new error-codes being defined for new PIB objects. In contrast, the SNMP message format has a single error-code, whose values are specified as part of the protocol-specification, and so can not be specific to individual objects, and new ones can not be specified in MIBs. (Note also that one of SNMP's design parameters, which is a fundamental aspect of many agent implementations, is that the varbindlist in an SNMP SetRequest is returned unchanged in the Response to that SetRequest; this design choice was made so that simple agents could modify a received message and send it as the Response, all in the same buffer.)

Fundamentally, COPS is an extensible protocol, so it will always be simpler to add functionality to COPS than SNMP. This will enable COPS to integrate the outsourcing and proxied functionality which will be required by AAA applications and bandwidth brokers.

3.9. Initiation by the PEP

For the purpose of obtaining policy, it is better for the device to locate a PDP, rather than for a PDP to try and find devices that need their policies. Initiation by the PEP works particularly well in the event of PDP failure with the primary/secondary mechanism (see above). It also works well in the event of a PEP reboot, in that the PEP can initiate communication as soon as it is ready rather than having to wait for the SNMP management station to poll it.

In contrast, SNMP is designed such that multiple management stations can monitor a device as and when they need to, and hence it is each management station that must initiate SNMP communication. It is also not clear how a "secondary" SNMP management station would constantly monitor for a failure of communication between the primary and each device. Thus, the (normal) SNMP model does not seem to lend itself to initiation by the PEP.

It has been suggested an alternative way to get initiation by the PEP using SNMP would be to have the device (e.g., a router) contain a command generator [SNMPARCH]. While routers do not usually contain command generators, it is possible that they could. However, the use of SNMP's get-type requests would be a very awkward means of obtaining only that subset of policies which are relevant to this router; such requests might well need an SQL-like query capability (or the similar searching capabilities of LDAP which some PDPs use to extract policies out of a Directory.)

With COPS, initiation by the PEP is achieved without the PEP needing to issue any search queries. Instead, on establishment of the TCP connection, the PEP sends an indication of its capabilities and status to the PDP. It then asks the PDP to send it the relevant policies.

3.10. Deleting Policy

COPS for Provisioning includes a mechanism in the protocol for specifying whether the policies in a DEC message are being installed or deleted. The deletion capability can remove a single row (called a "policy rule instance" in COPS), or all rows of a table (all "policy rule instances" of a "policy rule class").

SNMP can have a similar capability by having MIB objects which can be set to cause various instances to be deleted: RowStatus is common for deleting a particular row; objects to delete all rows in a table are not so common.

4. PIBs

4.1. Syntactic Differences

PIBs are intentionally similar to MIBs, in order to leverage knowledge of and experience with MIBs. The major syntactic differences are designed to accommodate the differences in COPS versus SNMP.

- a PIB module begins with keyword PIB-DEFINITIONS rather than the keyword DEFINITIONS, to identify it as a PIB rather than a MIB. This will avoid the confusion which was common when MIBs were used with ATM's ILMI protocol [ILMI].

- a POLICY-ACCESS clause in a PIB replaces the MAX-ACCESS clause in a MIB. This accommodates the different granularity of access (row versus columnar object), and the different message types in the two protocols. The POLICY-ACCESS clause is specified only on a PIB class (an SNMP "table"), whereas SNMP's MAX-ACCESS is specified on all MIB objects including columnar objects. The use of "notify" in the POLICY-ACCESS clause signifies a capability or status class whose vector of values must be passed in the initial REQ and when synchronizing state; the use of "install" signifies a class into which the PDP can download policies via a DEC message.

- a PIB includes an additional clause allowing per-class install errors to be specified.

- PIBs do not allow scalar objects (since atomic access to rows would cause complications in accessing scalars). Of course, MIBs for use in Policy Provisioning could be also be defined without scalars, but then there would be two types of MIBs - those that could be used with Policy Provisioning, and those that could not; this would no doubt cause problems through customers trying to use the ones that could not for Policy Provisioning.

- PIBs are a little simpler than MIBs, because they do not need to include multi-manager synchronization objects or objects for deleting one/more/all rows, such as RowStatus, spin-locks, etc., nor do they need to specify procedures for how these additional objects are used.

Note that at this point in time, the specification of the SoPI is not yet as complete as will be needed for it to be standardardized. Work is ongoing to rectify this.

4.2. Semantic Differences

At the semantic level, PIBs are aimed at the definition of abstracted, or "higher-level" policy, e.g., network-wide policy. Multiple types of higher-level/network-wide abstractions are expected to be defined in the future.

The specific example of this defined in the first PIB [QOSPIB] is the granularity at which policy configuration is specified for network interfaces. Specifically, this first PIB uses the concept of "role" applied to interfaces. This concept is explained in [POLICYTERMS] as follows:

> Roles can be used to identify specific objects (e.g., device interfaces) that should be configured in a common manner using one or more policies. These interfaces may be defined by the purpose that they play in the network (e.g., "edge" vs. "backbone"), the characteristics of the object (e.g., frame relay interfaces require a different configuration than ATM interfaces), or other factors.
>
> Roles provide a powerful abstraction mechanism. They enable new policies to be specified for a single role, and have them applied to the devices that use that role. This is much more efficient and less error prone than having to specify a new policy for each and every individual network component. In addition, it enables policies to be modified at the (single) role level, instead of having to search for every occurrence of every policy and individually modify the policy. But most importantly, it enables the devices and their interfaces to be abstracted from the Policy Server. In other words, the Policy Server no longer needs to have intimate knowledge of each and every device (let alone each and every device interface!) in the network.

All interface-related policies in the PIB are defined, not per individual interface, but on a per-role basis. So, PIBs do not have the interface-naming issues of the [IF-MIB], where the original concept of physical interfaces has been expanded with logical interfaces and dynamically-created/deleted interfaces, to the extent that three different ways of naming interfaces are needed (ifIndex, ifName and ifAlias).

In contrast, an SNMP MIB is naturally aimed at device-specific configuration and monitoring. Whereas the same policies can be applied to two similar (but not identical) interfaces having the same role, the MIB must allow each interface to have different status, different statistics and different low-level configuration. Therefore, a MIB is needed which contains per-interface information (normally indexed by ifIndex), irrespective of whether roles are used to define policies.

4.3. SNMP and MIBs are Still Needed

It is worth repeating for the sake of emphasis that the use of COPS and PIBs does not obviate the need for SNMP, which is still needed for

- monitoring, which is not a function of the COPS protocol, and

- setting local configuration.

As an example of local configuration, see the COPS Client MIB, [COPS-MIB] which has a read-create table for configuring the addresses of primary/secondary/etc. PDPs.

5. What-if Questions

5.1. Could SNMP be Modified ?

One might ask: why couldn't SNMP be modified to have the same functionalities as are described for COPS ? The answer is that with sufficient changes, it could. However, it would no longer be the same protocol, and this new different (and more complex) protocol would need to be an addition (not a replacement) for the existing protocol which currently (and successfully) provides the network monitoring functions of the Internet.

As well as the differences, there would, of course, be similarities. The code-sharing due to these similarities would seem to be the only advantage of such major modifications to the SNMP protocol. However, implementations of COPS for Provisioning would also result in code-sharing, since some of the COPS design choices are specifically aimed at this:

- the similarity of PIBs to MIBs,

- the use of OIDs for naming rows in PIBs,

- the use of the BER encoding for OID and data values in COPS messages.

This together with the code sharing between the multiple COPS Client-types could well produce a larger amount of code-sharing.

Further, there would be at least one disadvantage of having two different protocols both being called SNMP, fulfilling different functions - it would result in significant confusion in the Internet community. This is a huge issue; there is already so much fear, uncertainty and doubt with respect to SNMP evolution that more confusion would be a "real killer".

5.2. Could Roles be used in MIBs ?

One might ask: why couldn't roles be used in MIBs ? The answer is that they could, but to do so would result in having two different MIBs at different granularities. This is no better than having a MIB and a PIB at different granularities. At least with a MIB and a PIB, it is clear that one of them (and which one of them) is specifically defined to be at a higher-level for use by a protocol optimized for bulk configuration of multi-attribute objects.

Indeed, [QOSPIB] defines a mechanism for converting a PIB into a MIB. The derived MIB is specified as read-only, with the primary intention of use with existing MIB tools, and/or use by SNMP in monitoring the policies downloaded by COPS. Note that information is lost in the process of converting a PIB into a MIB, i.e., the conversion is not invertable to creating a PIB from a MIB. Nevertheless, all the policy configured in the PIB by COPS is accessible by SNMP via the derived MIB.

6. Security Considerations

Security mechanisms are defined for both SNMP (in [USEC] and [VACM]), and for COPS (in [COPS]). SNMP's mechanisms were defined for exclusive use by SNMP; this design decision was based (in part) on:

- SNMP must not rely on the mechanisms it is trying to fix; for example, from [RFC1270]:

  > ... SNMP must continue to operate (if at all possible) when the network is operating at its worst. For other applications, such as Telnet or FTP, the user can always "try again later" if the network is operating poorly. On the other hand, the major purpose of a network management protocol is to fix the network when it is operating poorly so the "try again later" strategy is useless.

- IP-only mechanisms were insufficient, because of existing usage of SNMP over multiple network-layer protocols: not only IP, but AppleTalk, IPX, etc. as well.

With the use of COPS being exclusively over TCP, COPS provides the following security choices:

- a (mandatory to implement) COPS-specific message integrity object, providing authentication and replay protection, but not encryption.

- IPSEC, providing authentication, replay protection and/or encryption; presumably sharing the same key distribution mechanism as other uses of IPSEC.

- session level security mechanisms (SSL/TLS), which provide increased security (and increased overhead!!).

Some of SNMP security's complications derive from the granularity of its authentication and authorization mechanisms:

- for authentication, SNMP provides per-user granularity.

- for access control, SNMP provides granularity to the object-level or instance-level, and to the types of allowed operations (read, write, etc.).

These low levels of granularity are necessary when multiple network managers are using multiple network management stations to access multiple MIBs. Their design accommodates, for example, access from an on-call network manager in the middle of the night to fix network problems, as well as out-of-band management. In contrast, these low-levels are not needed in the COPS model where the PDP has exclusive access to the PEP. So, COPS is only required to provide per-IP-host authentication, and per-session access control.

As a result, COPS security (including its configuration) is always simpler than SNMP's, and the reduced burden of configuring COPS security can sometimes be further reduced through its commonality with the security of other applications.

Furthermore, the deployment of SNMP security faces the difficulty of overcoming the inertia of the installed base of SNMP systems which lack security. The deployment of COPS security will be a simpler task since security is included in its initial (standardized) specification.

7. Acknowledgements

The authors thank the following individuals who provided comments on this document: David Durham, Silvano Gai, and John Seligson.

8. Intellectual Property

Director.

9. References

[COPS]
Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R., and A. Sastry, "The COPS (Common Open Policy Service) Protocol", draft-ietf-rap-cops-07.txt, work-in-progress, August 1999.

[COPS-PROV]
Reichmeyer, F., Herzog, S., Chan, K., Durham, D., Yavatkar, R., Gai, S., McCloghrie, K., and A. Smith, "COPS Usage for Policy Provisioning", draft-ietf-rap-pr-00.txt, work-in-progress, June 1999.

[COPS-RSVP]
Boyle, J., Cohen, R., Durham, D., Herzog, S., Rajan, R., and A. Sastry, "COPS usage for RSVP", draft-ietf-rap-cops-rsvp-05.txt, work-in-progress, June 1999.

[COPS-MIB]
Smith, A., Partain, D., and J. Seligson, "Definitions of Managed Objects for Common Open Policy Service (COPS) Protocol Clients", draft-ietf-rap-cops-client-mib-00.txt, work-in-progress, June 1999.

[DCLASS]
Bernet, Y., "Usage and Format of the DCLASS Object With RSVP Signaling", draft-ietf-issll-dclass-00.txt, work-in-progress, June 1999.

[DSARCH]
Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Service", RFC 2475, December 1998.

[DSRSVP]
Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Speer, M., Braden, R., and B. Davie, "Integrated Services Operation Over Diffserv Networks", draft-ietf-issll-diffserv-rsvp-02.txt, work-in-progress, June 1999.

[IF-MIB]
McCloghrie, K., and F. Kastenholz, "The Interfaces Group MIB

using SMIv2", RFC 2233, November 1997.

[ILMI]
ATM Forum Technical Committee, "Integrated Local Management Interface (ILMI) Specification", Version 4.0, af-ilmi-0065.000, September 1996.

[INTSERV].
Braden, R., Clark, D., and S. Shenker, "Integrated Services in the Internet Architecture: an Overview", RFC 1633, June 1994.

[POLICYTERMS]
Strassner, J., and E. Ellesson, "Terminology for describing network policy and service", draft-ietf-policy-terms-00.txt, work-in-progress, June 1999.

[QOSPIB]
Fine, M., McCloghrie, K., Seligson, J., Chan, K., Hahn, S., and A. Smith, "Quality of Service Policy Information Base", draft-mfine-cops-pib-01.txt, work-in-progress, June 1999.

[RFC1270]
Kastenholz, F., "SNMP Communications Services", RFC 1270, October 1991.

[RFC2026]
Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, BCP 9, October 1996.

[RSVP]
Braden, R., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, September 1997.

[SMIv2]
McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[TCv2]
McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[CONFv2]
McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[SNMPARCH]
Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2571, April 1999.

[SNMPTCP]
Schoenwaelder, J., "SNMP-over-TCP Transport Mapping", draft-irtf-nmrg-snmp-tcp-01.txt, work-in-progress, June 1999.

[USEC]
Blumenthal, U., and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2574, January 1998.

[VACM]
Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2575, January 1998.

10. Authors' Addresses

Keith McCloghrie
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
Phone: 408-526-5260
Email: kzm@cisco.com"

Michael Fine
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
Phone: 408-527-8218
Email: mfine@cisco.com"

11. Full Copyright Statement

Table of Contents