به نام خدا

ترجمه RFC 1157

درس مدیریت شبکه های کامپیوتری

استاد : خانم دکتر موحدی

دانشجو : آزاده دبیری

94725045

*بخش هایی که بنده ترجمه نموده ام :

(((1 و 2 و 3 و 3.1.4 و 1.3.1.4 و 4.1.4)))

1. وضعیت این یادداشت

این RFC انتشار مجدد RFC 1098، با تغییراتی در بخش " وضعیت این یادداشت " به علاوه اندکی اصلاحات تایپی است. این یادداشت یک پروتکل ساده را تعریف می کند، که کاربران از راه دور منطقی (منطقا راه دور) می توانند توسط آن اطلاعات مدیریتی برای یک عنصر شبکه را بررسی کرده و یا تغییر دهند. به خصوص، با یادداشت های همراه آن است که ساختار اطلاعات مدیریتی همراه با پایگاه داده اطلاعات مدیریتی توصیف می شود، این اسناد یک معماری و سیستم ساده و قابل اجرا برای مدیریت شبکه های مبتنی بر TCP / IP و به خصوص اینترنت فراهم می کنند.

انجمن فعالیت های اینترنتی توصیه می کند که تمام پیاده سازی های IP و TCP تحت کنترل شبکه درآیند. که به پیاده سازی MIB اینترنت (RFC-1156) و حداقل یکی از دو پروتکل مدیریتی توصیه شده ی SNMP (RFC-1157) و یا CMOT (RFC-1095) اشاره دارد. لازم به ذکر است که، هم اکنون، SNMP یک استاندارد کامل اینترنت و CMOT یک پیش نویس استاندارد است. همچنین میزبان و دروازه مورد نیاز RFC ها را برای اطلاعات مشخص تر (بیشتر) برای کاربرد این استاندارد ببینید.

لطفا برای کسب اطلاعات فعلی (به روز) درباره وضعیت و چگونگی پروتکل های استاندارد اینترنت به آخرین نسخه از RFC " استانداردهای پروتکل رسمی IAB" مراجعه کنید.
توزیع این یادداشت بدون محدودیت می باشد.

2. معرفی

همانطور که در RFC 1052 گزارش شده است، توصیه های IAB برای توسعه استانداردهای مدیریتی شبکه اینترنت [1]، یک استراتژی دو لبه برای مدیریت شبکه های مبتنی بر TCP /IP را تضمین کرده بود. در کوتاه مدت، پروتکل مدیریت شبکه ساده (SNMP) برای مدیریت گره ها در مجموعه ی اینترنت مورد استفاده قرار می گرفت. در بلند مدت، استفاده از چارچوب مدیریت شبکه OSI مورد بررسی قرار گرفت. دو سند برای تعریف اطلاعات مدیریتی تولید شده بودند : RFC 1065، معرف ساختار اطلاعات مدیریتی (SMI) [2]، و RFC 1066، معرف پایگاه اطلاعات مدیریتی (MIB) [3]. این اسناد با هر دو چارچوب مدیریت شبکه SNMP و OSI به صورت سازگار طراحی شده بودند.

این استراتژی در کوتاه مدت کاملا موفق بود: فن آوری مدیریت شبکه ی اینترنت، در عرض چند ماه توسط جوامع تحقیقاتی و تجاری به میدان فرستاده شد. در نتیجه، بخشی از جامعه اینترنت به موقع به شبکه ی قابل مدیریت تبدیل شدند.

همانطور که در RFC 1109 گزارش شده است، گزارشی از گروه دوم نقد و بررسی مدیریت شبکه موردی، الزامات چارچوب های مدیریت شبکه SNMP و OSI بسیار با پیش بینی متفاوت بود. به این ترتیب، نیاز به سازگاری بین SMI / MIB و هر دو چارچوب متوقف شد. این عمل به چارچوب عملیاتی مدیریت شبکه ، SNMP، اجازه داد تا به نیازهای عملیاتی جدید در جامعه اینترنت با تولید اسناد تعریف آیتم های جدید MIB پاسخ دهد.

IAB برای اینکه SNMP، SMI، و MIB اولیه اینترنت، "پروتکل های استاندارد" کاملی شوند، آنها را با وضعیت "توصیه شده" تعیین کرده است. با این کار، IAB توصیه میکند که همه ی پیاده سازی های IP و TCP دارای قابلیت مدیریت شبکه شوند و از آنجا که پیاده سازی ها با قابلیت مدیریت شبکه هستنند، انتظار می رود با SMI، MIB و SNMP سازگار بوده و آنها را اجرا کنند.

به این ترتیب، چارچوب مدیریت شبکه فعلی برای شبکه های مبتنی برTCP/IP شامل موارد زیر است: ساختار و تعریف اطلاعات مدیریتی برای شبکه های مبتنی برTCP/IP، که چگونگی تعریف اشیاء مدیریتی موجود در MIB را شرح می دهد، مندرج درRFC 1155 [5]؛ پایگاه اطلاعات مدیریتی برای مدیریت شبکه های مبتنی برTCP/IP، که اشیاء مدیریتی موجود در MIB را توصیف می-کند، مندرج در RFC 1156 [6]؛ و پروتکل مدیریت شبکه ساده، که پروتکل مورد استفاده برای مدیریت این اشیاء را معرفی می کند، مندرج در این یادداشت.

همانطور که در RFC 1052 گزارش شده است، توصیه های IAB برای توسعه استانداردهای مدیریت شبکه اینترنت [1]، انجمن فعالیت های اینترنتی به نیروی کار مهندسی اینترنت (IETF) دستور داد تا دو کار گروه جدید در حوزه مدیریت شبکه ایجاد کند. یک گروه به تشخیص و تعریف عناصر بیشتر موظف شد تا در پایگاه اطلاعات مدیریتی (MIB) گنجانده شوند. گروه دیگر به تعریف تغییرات در پروتکل مدیریت شبکه ساده (SNMP)، برای تطبیق نیازهای کوتاه مدت جوامع عملیاتی و تجاری شبکه، و هم تراز کردن با خروجی کار گروه MIB موظف شد.

گروه کاری MIB دو یادداشت ارائه داد، یکی معرف یک ساختار اطلاعات مدیریتی (SMI) [2] برای استفاده توسط اشیاء مدیریتی موجود در MIB است. یادداشت دوم [3]، لیستی از اشیاء مدیریتی تعریف می کند.

خروجی کار گروه توسعه SNMP این یادداشت است، که شامل تغییرات تعریف اولیه SNMP مورد نیاز برای رسیدن به هم ترازی با خروجی کار گروه MIB است. تغییرات باید به منظور سازگاری با بخشنامه IAB که کار گروه ها باید " ا لزاما به ساده نگه داشتن

SNMP بسیار حساس باشند"، حداقل باشند. اگر چه توجه و بحث زیادی درباره تغییرات SNMP وجود دارد که در این یادداشت نیز منعکس شده، پروتکل نتیجه با پروتکل پیشینش، پروتکل مانیتورینگ دروازه ساده (SGMP) [8]، چندان ناسازگار هم نیست . اگرچه نحو (syntax) پروتکل تغییر یافته است، فلسفه اصلی، تصمیم گیری های طراحی، و معماری دست نخورده باقی ماند. به منظور جلوگیری از سردرگمی، پورت UDP جدیدی برای استفاده توسط پروتکل شرح داده شده در این یادداشت اختصاص داده شده است.

3. معماری SNMP

به طور ضمنی در مدل معماری SNMP مجموعه ای از ایستگاه های مدیریت شبکه و عناصر شبکه موجود است. ایستگاه های مدیریت شبکه برنامه های کاربردی مدیریتی که نظارت و کنترل عناصر شبکه می باشند را اجرا می کنند. عناصر شبکه دستگاه هایی مانند میزبان، دروازه، سرور ترمینال، و مانند آن هستند، که عوامل مدیریتی مسئول انجام توابع مدیریت شبکه ی درخواستی توسط ایستگاه-های مدیریت شبکه می باشد. پروتکل مدیریت شبکه ساده (SNMP) برای برقراری ارتباط اطلاعات مدیریتی بین ایستگاه های مدیریت شبکه و عوامل موجود در عناصر شبکه استفاده شده است.

1.3. اهداف معماری

SNMP آشکارا تعداد و پیچیدگی توابعی را که از سوی عامل های مدیریتی قابل درک اند، کمینه می‌کند. این هدف حداقل در چهار جنبه‌ی زیر خلاصه می‌شود:

1. هزینه‌ی گسترش نرم‌افزار عامل مدیریتی مورد نیاز برای پشتیبانی از پروتکل، کاهش می‌یابد.

2. از آن‌جا که سطح(degree) تابع مدیریتی که از راه دور پشتیبانی می‌شود، افزایش یافته، در نتیجه استفاده‌ی کامل از منابع اینترنت را در اختیار مدیریت وظایف قرار می‌دهد.

3.از آن‌جا که سطح(degree) تابع مدیریتی که از راه دور پشتیبانی می‌شود، افزایش یافته است؛ به دنبال آن محدودیت‌های ممکن کم‌تری بر روی قالب و پیچیدگی ابزارهای مدیریتی اعمال می‌کند.

4. مجموعه‌ی ساده شده‌ای از توابع مدیریتی را که به راحتی قابل فهم هستند و توسط گسترش‌دهندگان ابزارهای مدیریت شبکه مورد استفاده قرار می‌گیرند، معرفی می‌کنند.

هدف دوم پروتکل این است که نمونه‌ی عملی برای کنترل و مانیتور کردن کافی جهت تطبیق با جنبه‌های غیرقابل پیش‌بینی و افزونه-ی عملگر و مدیریت شبکه، قابل گسترش باشد.

هدف سوم این است که معماری تا جای ممکن مستقل از معماری و مکانیزم host و یا gateway خاصی از شبکه باشد.

2.3 عناصر معماری

معماری SNMP راه حلی را برای مسئله مدیریت شبکه درباره‌ی موارد زیر به وضوح بیان کرده است:

1. وسعت اطلاعات مدیریتی که با پروتکل در ارتباط هستند.

2. نمایش اطلاعات مدیریتی که با پروتکل در ارتباط هستند.

3. عملیات بر روی اطلاعات مدیریتی که توسط پروتکل پشتیبانی می‌شوند.

4. نحوه و معنای تغییرات و تبادل بین موجودیت‌های مدیریتی.

5. تعریف روابط اجرایی بین موجودیت‌های مدیریتی.

6. نحوه و معنای ارجاعات به اطلاعات مدیریتی.

1.2.3 وسعت اطلاعات مدیریتی

وسعت این اطلاعات دقیقا همان نمونه‌های انواع داده‌ای غیرمجتمع است که در استاندارد اینترنت MIB و یا هر جای دیگر بنا بر مجموعه قراردادهای موجود در استاندارد اینترنت SMI ارائه شده است.

جهت پشتیبانی از انواع داده‌ای مجتمع شده در MIB نه نیازی به مطابقت با SMI است و نه این که توسط SNMP تشخیص داده شود.

2.2.3 نمایش اطلاعات مدیریتی

اطلاعات مدیریتی مرتبط با عملیات SNMP بر اساس زیرمجموعه‌ای از زبان ASN.1 نمایش داده می‌شوند که برای نمایش انواع داده-ای غیرمجتمع SMI تعریف شده است.

SGMP قراردادهای موردنیاز جهت استفاده از زیرمجموعه‌ای خوش تعریف از زبان ASN.1 را، پذیرفته است. SNMP این سنت را با بهره‌برداری از زیرمجموعه‌ای کمی پیچیده‌تر از ASN.1 جهت تعریف managed objectها و واحدهای داده‌ای پروتکل برای آن objectها، ادامه داده و مورد گسترش قرار دهد است. همچنین علاقه به سهولت گذار احتمالی به پروتکل های مدیریت شبکه بر پایه-ی OSI، منجر به تعریف ساختار استاندارد اینترنت اطلاعات مدیریتی (SMI) و پایگاه اطلاعات مدیریتی (MIB) به زبان ASN.1 شده است. استفاده از ASN.1 به دلیل موفقیت آن در تلاش‌های اولیه،به عنوان مثال SGMP با استقبال بیش تری مواجه شد.

محدودیت در استفاده از ASN.1 که بخشی از SMI است، موجب حمایت ساده و معتبر بودن آن در آزمایشات SGMP شد.

همچنین به منظور سادگی، SNMP فقط از زیرمجموعه‌ای از قوانین پایه‌ای کد کردن در زبان ASN.1 استفاده می‌کند. به عنوان مثال همه‌ی کدها از یک قالب با طول مشخص استفاده می‌کنند. همچنین در مواقع مجاز ، کدگذاری‌های غیر ساختمند به جای کدگذاری-

های ساختمند، استفاده می‌شوند. این محدودیت بر روی همه‌ی جنبه‌های کدگذاری ASN.1 اعمال می‌شود. هم برای واحدهای داده‌ای پروتکل در سطح بالا و هم انواع داده‌ایی که شامل می‌شوند.

3.2.3 عملیات بر روی اطلاعات مدیریتی

SNMP همه ی توابع مدیریتی را به عنوان تغییرات و بازرسی‌های متغیرها مدل می‌کند. بنابراین یک موجودیت پروتکل که بر روی میزبان منطقا از راه دور قرار دارد. (احتمالا همان خود عنصر شبکه) جهت بازیابی(get) و یا تغییر (set) متغیرها با عامل مدیریتی که بر روی عناصر شبکه قرار دارند، تعامل می‌کنند. این استراتژی حداقل دو پیامد زیر را در پی دارد:

1. یکی از پیامدها این است که تعداد توابع مدیریتی مهم که توسط عامل قابل درک اند، تنها محدود به دو عملات می‌شود: یک عملگر برای مقدار دادن به یک configuration خاص و یا پارامترهای دیگر و عملگر دیگری جهت بازیابی آن‌ها.

2. پیامد دیگر این تصمیم این است که از دستورات مدیریتی ضروری(imperative) می‌پرهیزد: تعداد این نوع توابع عملا همواره در حال افزایش است و معنای این دستورات نیز پیچیدگی‌های متنوعی دارد.

استراتژی نهفته در SNMP همان مانیتور کردن وضعیت شبکه است که در سطح علمی از جزئیات که اصولا با رای‌گیری جهت تایین اطلاعات مناسب آن بخش مرکز کنترل، انجام می‌شود. تعداد محدودی از پیام‌های ناخواسته یا همان trap، زمان‌بندی و کانون رای-گیری را هدایت می‌کنند. تعداد محدود trap با هدف سادگی و کمینه کردن ترافیکی توابع مدیریت شبکه تولید می‌کنند، سازگاری دارد.

منع دستورات ضروری از طرف مجموعه توابع مدیریتی، غیر محتمل است که از هر عملیات دلخواه عامل مدیریتی جلوگیری کند. در حال حاضر، بیش‌تر دستورات جهت مقداردهی کردن پارامترها و یا بازیابی آن‌ها هستند و در این مدل مدیریتی، تابعی با دستورات ضروری کم تعداد، که در حال حاضر پشتیبانی می‌شود. به راحتی در مد آسنکرون قابل اجراست.

در این شما، یک دستور ضروری می‌تواند به عنوان مقداردهی کردن یک پارامتر که متعاقبا یک عمل دلخواه را راه‌اندازی می‌کند، در نظر گرفته شود. به عنوان مثال، به جای پیاده سازی دستور reboot این عمل به راحتی با مقداردهی کردن پارامتری که ثانیه‌های باقی‌مانده تا reboot شدن سیستم را نشان می‌دهد، قابل راه‌اندازی است.

4.2.3 قالب و معنای تبادلات پروتکل

مخابره‌ی اطلاعات مدیریتی بین موجودیت‌های مدیریتی در SNMP در قالب تبادل پیغام‌های پروتکل انجام می‌شود. قالب و معنای این پیغام‌ها در بخش 4 توضیح داده شده است.

جهت سازگاری با هدف کمینه کردن پیچیدگی عامل مدیریتی، تبادل پیغام‌های SNMP فقط به یک سرویس unreliable datagram نیاز است و هر پیغام به طور کامل و مستقل در یک datagram انتقال نمایش داده می‌شود.

درحالی که این سند،تبادل پیام‌ها را از طریق پروتکل UDP در نظر گرفته است؛ مکانیزم SNMP به طور کلی مناسب استفاده در سرویس‌های transport متنوعی است.

5.2.3 تعریف روابط اجرایی

معماری SNMP تنوع روابط اجرایی میان موجودیت هایی را که در پروتکل شرکت می کنند، می پذیرد. موجودیت های مستقر در ایستگاه های مدیریتی و عناصر شبکه که با استفاده از SNMP با یکدیگر ارتباط دارند، موجودیت های کاربردی SNMP نامیده می شوند. پردازش های نظیر که SNMP را پیاده سازی می کنند و از موجودیت های کاربردی SNMP پشتیبانی می کنند، موجودیت های پروتکل نامیده می شوند.

دسته دوتایی متشکل از "یک عامل SNMP" و "مجموعه دلخواهی از موجودیت های کاربردی SNMP"، اجتماع SNMP نام دارد. هر اجتماع SNMP با یک رشته هشت تایی(octet) نام گذاری می شود که اسم اجتماع گفته می شود.

پیام SNMP که توسط یک موجودیت کاربردی SNMP تولید می شود و این موجودیت به اجتماع SNMP تعلق دارد، پیام معتبر SNMP نامیده می شود. مجموعه قوانینی که توسط این پیام برای یک اجتماع SNMP خاص مشخص می شود، شمای احراز هویت نام دارد. پیاده سازی تابعی که پیام های معتبر SNMP را مطابق با طرح های احراز هویت شناسایی کند، سرویس احراز هویت نام دارد.

قطعاً مدیریت موثر روابط بین موجودیتهای کاربردی به سرویس های احراز هویت نیاز دارد که (با استفاده از رمزنگاری یا تکنیک های دیگر) این سرویس ها قادرند پیام های معتبر SNMP را با درجه بالایی از اطمینان شناسایی کنند. برخی پیاده سازی های SNMP تنها پشتیبانی یک سرویس جزئی احراز هویت که تمام پیام های SNMP را به عنوان پیام های معتبر شناسایی می کند، نیاز دارند.

برای هر مولفه شبکه، یک زیرمجموعه از اشیا در MIB وجود دارد که متعلق به آن می باشد و دید SNMP MIB نام دارد. دقت کنید که اسامی انواع شی موجود در دید SNMP MIB نیازی نیست که به یک زیردرخت از فضای نام نوع شی تعلق داشته باشد.

یک عنصر از مجموعه {READ-ONLY, READ-WRITE}، مد دسترسی SNMP نام دارد.

دسته دوتایی "مد دسترسی SNMP" و "دید SNMP MIB" ، پروفایل اجتماع SNMP نامیده می شود. پروفایل اجتماع SNMP نوع دسترسی به متغیرهای موجود در دید MIB مشخص شده را نشان می دهد. در یک پروفایل اجتماع SNMP، دسترسی به هر متغیر موجود در دید MIB مطابق با قراردادهای زیر نشان داده می شود.

(1) اگر متغیر ذکر شده با دسترسی "none" در MIB تعریف شده باشد آنگاه برای هر عملگر یک عملوند غیر قابل دسترس خواهد بود.

(2) اگر متغیر ذکر شده با دسترسی "read-write" یا "write-only" در MIB تعریف شده باشد و مد دسترسی پروفایل داده شده READ-WRITE باشد، آنگاه این متغیر به عنوان یک عملوند برای عملیات get و set و trap در دسترس خواهد بود.

(3) درغیر این صورت، متغیر به عنوان یک عملوند برای عملیات get و trap در دسترس می باشد.

(4) در مواردی که متغیر"write-only" عملوند مورد استفاده در عملیات get و trap باشد، مقدار داده شده برای متغیر implementation-specific است.

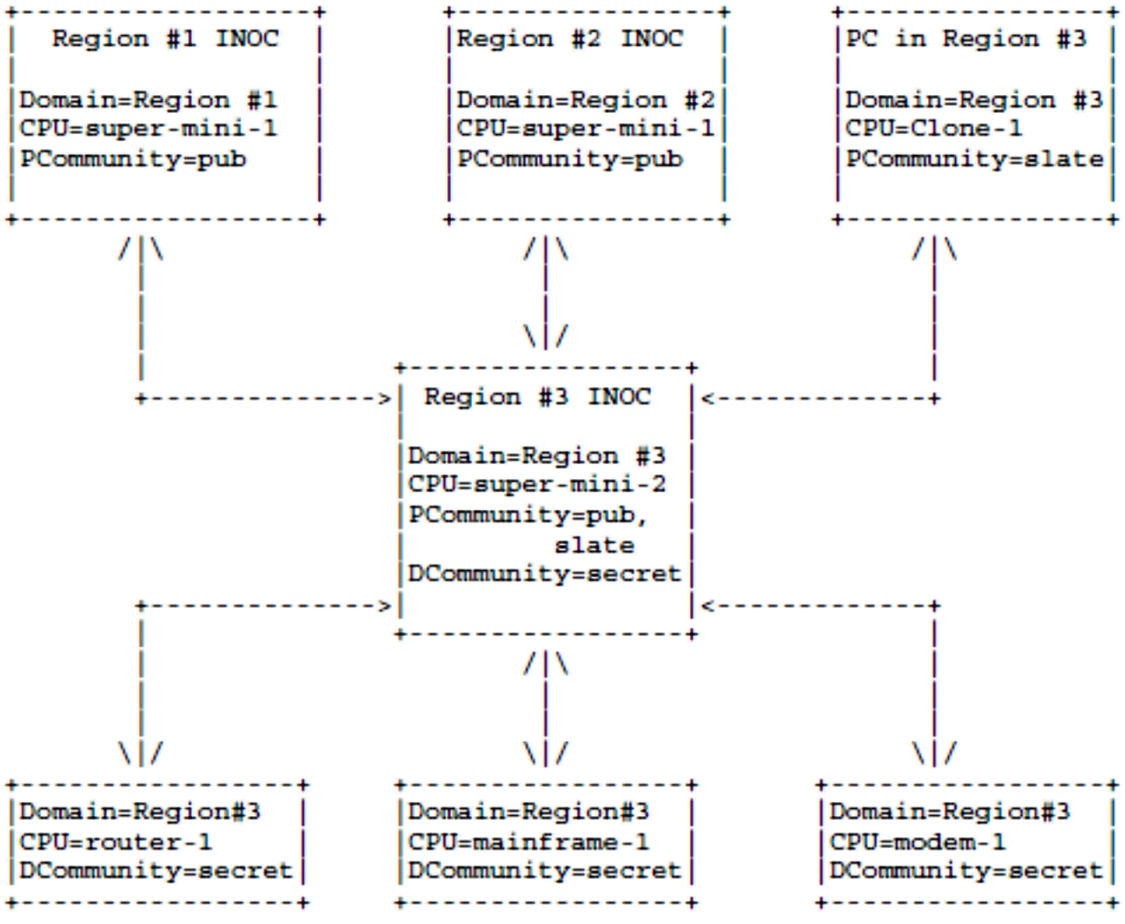
دسته دوتایی "اجتماع SNMP" و "پروفایل اجتماع SNMP"، سیاست دسترسی SNMP نامیده می شود. یک سیاست دسترسی، پروفایل اجتماع خاصی را که توسط عامل SNMP و از یک اجتماع SNMP برای اعضای دیگر آن اجتماع تهیه شده است، نشان می دهد. همه روابط اجرایی بین موجودیت های کاربردی SNMP در جملات و اصطلاحات سیاستهای دسترسی SNMP تعریف می شوند.

برای هر سیاست دسترسی SNMP، اگر عنصر شبکه روی عاملی که در اجتماع SNMP خاصی قرار دارد همان عنصر مربوط به دید MIB برای پروفایل خاصی نباشد، آن سیاست دسترسی؛ proxy نامیده می شود. عامل SNMP که به یک سیاست دسترسی proxy مرتبط است عامل SNMP proxy نام دارد. همانطور که تعریف بی دقت از سیاست های دسترسی proxy می تواند در چرخه های مدیریتی اثر بگذارد، تعریف محتاطانه سیاست های proxy در حداقل دو مورد مفید است:

(1) اجازه نظارت و کنترل عناصر شبکه را می دهد که با استفاده از پروتکل مدیریت و پروتکل انتقال نشان ناپذیر هستند. یک عامل proxy ممکن است یک تابع تبدیل پروتکل را فراهم کند که به یک ایستگاه مدیریتی مجوز می دهد تا یک قالب مدیریتی سازگار را به همه عناصر شبکه، دستگاه های ورودی مثل مودم ها،مالتی پلکسورها و دستگاه های دیگر که قالبهای مدیریتی متفاوتی را پشتیبانی می کنند، اعمال کند.

(2) عناصر شبکه را از سیاست های کنترل دسترسی محفوظ نگه می دارد. برای مثال، یک عامل proxy ممکن است کنترل دسترسی پیچیده ای را پیاده سازی کند که بدون اینکه پیچیدگی عنصر شبکه افزایش یابد، بوسیله آن زیرمجموعه گوناگونی از متغیرهای داخل MIB برای ایستگاه های مختلف دسترس پذیر ساخته شوند .

شکل 1 ارتباط بین ایستگاههای مدیریتی، عاملهای proxy وعامل های مدیریتی را نشان می دهد. در این مثال، عامل proxy یک مرکز عملیات شبکه اینترنت(INOC) برای یک حوزه اجرایی که ارتباط استانداردی با یک مجموعه از عاملهای مدیریتی دارد، تصور شده است.



(شکل1. مثالی برای پیکربندی مدیریت شبکه)

Domain: حوزه اجرایی یک عنصر

PCommunity: نام یک اجتماع که از عامل proxy بهره می برد.

DCommunity: نام یک اجتماع مستقیم

6.2.3 شکل و معنی مرجع برای اشیای مدیریت شده

SMI به تعریف یک آدرس پروتکل مدیریتی نیاز دارد که موارد زیر را دربربگیرد:

(1) تفکیک منابع مبهم MIB

(2) تفکیک منابع MIB در حضور نسخه های مختلف MIB، و

(3) شناسایی نمونه های خاص انواع شی تعریف شده در MIB

1.6.2.3 تفکیک منابع مبهم MIB

به دلیل اینکه حوزه هر عمل SNMP به شکل مفهومی به اشیای مربوط به یک عنصر شبکه محدود شده است و چون تمام منابع SNMP برای اشیای MIB با اسامی واحدی(به صورت ضمنی یا صریح) هستند، احتمال اینکه هر منبع SNMP برای هر نوع شی تعریف شده در MIB بتواند به چندین نمونه از آن نوع اشاره کند، وجود ندارد.

2.6.2.3 تفکیک منابع موجود در نسخه های MIB

نمونه شی ارجاع داده شده به هر عمل SNMP مستقیماً به عنوان قسمتی از درخواست عمل یا(در مورد یک عمل get-next) بلافاصله بعدی اش در MIB مشخص می شود. به طور خاص یک منبع برای یک شی به عنوان قسمتی از نسخه MIB استاندارد اینترنت به هر شی که قسمتی از نسخه گفته شده نیست اشاره نمی کند، جز در مواردی که عمل درخواست شده get-next باشد و نام شی مشخص شده آخر اسامی همه اشیای نشان داده شده به عنوان قسمتی از نسخه گفته شده از MIB استاندارد اینترنت باشد.!

3. 2. 6. 3 شناسایی نمونه های اشیاء

اسامی تمام انواع اشیاء در MIB ، یا در استاندارد اینترنت MIB ، یا در دیگر اسنادی که با قرارداد نام گزاری SMI مطابقت دارند، صریحا تعریف شده اند. SMI نیازمند آن است که پروتکل های مدیریتی منطبق، مکانیزم هایی برای شناسایی نمونه های منحصر به فرد از آن انواع اشیاء ،برای یک عنصر خاص شبکه تعریف کنند.

هر نمونه از انواع اشیاء که در MIB تعریف شده در عملیات SNMP ، با یک نام یکتا به اسم "نام متغیر" شناسایی میشود. به طور کلی نام یک منغیر SNMP ، یک شناسه شئ به فرم X.Y است که X نام یک نوع شئ غیر تجمعی میباشد که در MIB تعریف شده است و Y یک قطعه شناسه شئ میباشد که به شیوه ای خاص برای نوع شئ، نمونه مورد نظر را شناسایی میکند.

این استراتژی نام گزاری، کاملترین بهره گیری از معانی GetNextRequest-PDU (به بخش 4 مراجعه شود) را میپذیرد، چون نام هایی را به متغیر های مربوطه اختصاص میدهد که بر اساس آن پیوستگی در ترتیب تمام اسامی متغیر های شناخته شده در MIB حفظ میشود.

نام گزاری مبتنی بر نوع نمونه های اشیاء در ادامه برای تعدادی از کلاس های انواع اشیاء تعریف شده است. نمونه هایی از یک نوع شئ، که هیچ کدام از قرارداد های نام گزاری ذیل که مناسب نمیباشند، با شناسه های نام گزاری اشیاء به فرم X.O نام گزاری شده اند، که X نام نوع شئ مذکور در تعریف MIB میبباشد.

برای مثال، فرض کنید یک نفر میخواهد یک نمونه متغیر sysDescr را شناسایی کند، کلاس شئ برای sysDescr ، به فرم زیر میباشد:

Iso org dod internet mgmt. mib system sysDescr

1 3 6 1 2 1 1 1

از این رو، نوع شئ X، به صورت زیر خواهد بود:

1.3.6.1.2.1.1.1

که یک زیر شناسه ی نمونه 0. به آن اضافه میشود، که به فرم 1.3.6.1.2.1.1.1.0 ، تنها نمونه sysDescr را شناسایی میکند.

3. 2. 6. 1. اسامی انواع اشیاء ifTable

نام یک رابط زیر شبکه، S ، مقدار شناسه شئ به فرم i میباشد، که i مقدار آن نمونه از نوع شئ ifIndex مرتبط با S را دارا میباشد.

برای هر نوع شئ، t ، که اسم تعریف شده، n، یک پیشوند ifEntry دارد، یک نمونه، i ، از t ، با استفاده از یک شناسه شئ به فرم n.s نامگذاری شده است، که s نام رابط زیرشبکه ای است که i اطلاعات آن را ارائه میدهد.

برای مثال فرض کنید یک نفر میخواهد نمونه ای از متغیر ifType مرتبط با رابط 2 را شناسایی کند. بر این اساس ifType.2 نمونه مورد نظر را شناسایی خواهد کرد.

3. 2. 6. 2. اسامی انواع اشیاء atTable

نام یک آدرس شبکه ذخیره شده AT (AT-cached network address)، x ، یک شناسه شئ به فرم 1.a.b.c.d میباشد که a.b.c.d مقدار (در نشانه گذاری نقطه ای شناخته شده) atNetAddress برای نوع شئ مرتبط با X میباشد.

نام ترجمه ی آدرس معادل e ، یک مقدار شناسه شئ به فرم S.W میباشد، که S مقدار نمونه ی نوع شئ atIndex مرتبط با e میباشد و w نام آدرس شبکه ذخیره شده AT (AT-cached network address) مرتبط با e میباشد.

برای هر نوع شئ ،t ، که برای آن نام تعریف شده، n ، یک پیشوند از atEntry دارد، یک نمونه ، i ، از t با استفاده از شناسه شئ به فرم n.y که Y نام معادل ترجمه آدرسی میباشد که i درباره آن اطلاعات میدهد.

برای مثال، فرض کنی یک نفر میخواهد آدرس فیزیکی یک ورودی در جدول ترجمه آدرس (ARP cache) مرتبط با یک آدرس IP 89.1.1.42 با رابط 3 را پیدا کند. بر این اساس ، عبارت زیر:

atPhysAddress.3.1.89.1.1.42

نمونه مورد نظر را شناسایی خواهد کرد.

3. 3. 6. 2. اسامی انواع اشیاء ipAddrTable

نام یک عنصر شبکه با قابلیت آدرس دهی مبتنی بر IP، X، شناسه شئ به فرم a.b.c.d که a.b.c.d مقدار (در نشانه گذاری نقطه ای شناخته شده) آن نمونه از نوع شئ ipAdEntAddr مرتبط با x میباشد.

برای هر نوع شئ t، ، که برای آن نام تعریف شده، n ، یک پیشوند از ipAddrEntry دارد، یک نمونه ، i ، از t با استفاده از شناسه شئ به فرم n.y که Y نام عنصر شبکه با قابلیت آدرس دهی مبتنی بر IP میباشد که i درباره آن اطلاعات میدهد.

برای مثال، فرض کنی یک نفر میخواهد ماسک شبکه (Network Mask) یک ورودی در جدول رابط IP مرتبط با یک آدرس IP 89.1.1.42 را پیدا کند. براین اساس عبارت زیر:

ipAdEntNetMask.89.1.1.42

نمونه مورد نظر را شناسایی خواهد کرد.

3. 2. 6. 3. 4 اسامی انواع اشیاء ipRoutingTable

نام یک مسیر IP، X ، شناسه شئ به فرم a.b.c.d میباشد که a.b.c.d مقدار (در نشانه گذاری نقطه ای شناخته شده) آن نمونه از نوع شئ ipRoutingTable مرتبط با x میباشد.

برای هر نوع شئ t، ، که برای آن نام تعریف شده، n ، یک پیشوند از ipRoutingEntry دارد، یک نمونه ، i ، از t با استفاده از شناسه شئ به فرم n.y که Y نام مسیر IP میباشد که i درباره آن اطلاعات میدهد.

برای مثال، فرض کنی یک نفر میخواهد پرش بعدی یک ورودی در جدول مسیریابی IP مرتبط با مقصد 89.1.1.42 را پیدا کند.

براین اساس عبارت زیر:

ipRouteNextHop.89.1.1.42

نمونه مورد نظر را شناسایی خواهد کرد.

3. 2. 6. 3. 5 اسامی انواع اشیاء tcpConnTable

نام یک ارتباط TCP، X ، شناسه شئ به فرم a.b.c.d.e.f.g.h.i.j میباشد که a.b.c.d مقدار (در نشانه گذاری نقطه ای شناخته شده) آن نمونه از نوع شئ tcpConnLocalAddressمرتبط با x میباشد، که f.g.h.i مقدار (در نشانه گذاری نقطه ای شناخته شده) آن نمونه از نوع شئ tcpConnRemoteAddress مرتبط با x میباشد، که e مقدار آن نمونه از نوع شئ tcpConnLocalPort مرتبط با x میباشد، و j مقدار نمونه از نوع شئ tcpConnLocalPort مرتبط با x میباشد.

برای هر نوع شئ t، ، که برای آن نام تعریف شده، n ، یک پیشوند از tcpConnEntry دارد، یک نمونه ، i ، از t با استفاده از شناسه شئ به فرم n.y که Y نام ارتباط TCP میباشد که i درباره آن اطلاعات میدهد.

برای مثال، فرض کنی یک نفر میخواهد وضعیت یک ارتباط TCP بین آدرس محلی 89.1.1.42 روی پورت TCP 21 دوردست 10.0.0.51 روی پورت 2059 را بیابد.

براین اساس عبارت زیر:

tcpConnState.89.1.1.42.21.10.0.0.51.2059

نمونه مورد نظر را شناسایی خواهد کرد.

3. 2 .6 .3 .6 اسامی انواع اشیاء egpNeighTable

نام یک همسایه EGP ، x، یک شناسه شئ به فرم a.b.c.d میباشد که a.b.c.d مقدار (در نشانه گذاری نقطه ای شناخته شده) آن نمونه از نوع شئ egpNeighAddr مرتبط با x میباشد.

برای هر نوع شئ ،t ، که برای آن نام تعریف شده، n ، یک پیشوند از egpNeighEntry دارد، یک نمونه ، i ، از t با استفاده از شناسه شئ به فرم n.y که Y نام همسایه EGP میباشد که i درباره آن اطلاعات میدهد.

برای مثال، فرض کنی یک نفر میخواهد وضعیت همسایگی برای آدرس IP 89.1.1.42 را پیدا کند.

براین اساس عبارت زیر:

egpNeighState.89.1.1.42

نمونه مورد نظر را شناسایی خواهد کرد.

4. خصوصیات پروتکل

پروتکل مدیریت شبکه، یک پروتکل لایه‌ی application است که به وسیله‌ی آن می‌توان متغیرهای موجود در MIB عامل‌ها را تغییر داد و یا بازبینی نمود. ارتباط بین موجودیت‌های پروتکل با تبادل پیغام انجام می‌شود که هر پیغام به شکل مستقل و کامل در یک UDP datagram با استفاده از قوانین کدگذاری پایه‌ای ASN.1 نمایش داده می‌شوند.(همان‌طور که در بخش 2.2.3 بحث شد.) یک پیغام شامل یک شناسه‌ی version، یک نام گروه SNMP ، یک واحد داده‌ی پروتکل (PDU) است. یک موجودیت پروتکل پیغام‌ها را از طریق UDP Port 161 بر روی میزبان که همه‌ی پیام‌ها را دریافت می‌کند. به جز آن‌هایی که trap ها را گزارش می-کنند. پیام‌هایی که trapها را گزارش می‌کنند باید یک پیاده سازی از این پروتکل نیازی ندارد که پیغام‌هایی با طول بیش‌تر از 484 octet را بپذیرد. به هر حال ، توصیه شده است که پیاده سازی‌ها، datagramهایی با طول بیشتر را هم پشتیبانی کنند.

پشتیبانی از پنج PDU زیر در پیاده‌سازی پروتکل SNMP اجباری است:

GetRequest-PDU

GetNextRequest-PDU

GetResponse-PDU

SetResponse-PDU

Trap-PDU

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
FROM RFC1155-SMI;
-- top-level message
Message ::=
SEQUENCE {
version -- version-1 for this RFC
INTEGER {
version-1(0)
},
community -- community name
OCTET STRING,

data -- e.g., PDUs if trivial

ANY -- authentication is being used
}
-- protocol data units
PDUs ::=
CHOICE {
get-request
GetRequest-PDU,
get-next-request
GetNextRequest-PDU,

get-response
GetResponse-PDU,
set-request
SetRequest-PDU,
trap
Trap-PDU
}
-- the individual PDUs and commonly used
-- data types will be defined later
END
```

1.4 عناصر جهت پیاده‌سازی عملیات

در این بخش عملیات یک موجودیت پروتکل جهت پیاده سازی SNMP ، شرح داده شده است. به هر حال توجه داشته باشید که هدف محدود کردن معماری داخلی پیاده‌سازی‌ها نیست.

در متنی که پیش رو دارید واژه‌ی آدرس انتقال (transport address) استفاده شده است، که این واژه در UDP شامل یک آدرس IP پورت UDP است و در سرویس‌های انتقال دیگر برای پشتیبانی از SNMP استفاده می‌شود. که در این موارد تعریف آدرس انتقال باید مشخص شود.

عملیات سطح بالای موجودیت پروتکل که یک پیغام را ارسال می‌کند، به شرح زیر است:

1.ابتدا PDU مناسب را تولید می‌کند. به عنوان مثال GetRequest-PDU به عنوان یک ASN.1 object.

2. سپس این ASN.1 object را به همراه یک نام گروه و آدرس انتقال مبدا و مقصد آن، به سرویسی که شمای احراز هویت مورد نظر را پیاده‌سازی می‌کند، ارسال می کند. این سرویس احراز هویت یک ASN.1 object دیگر را برمی‌گرداند.

3.پس از آن موجودیت پروتکل با استفاده از نام گروه و ASN.1 object حاصل از مرحله‌ی قبل، یک ASN.1 message object می‌سازد.

4. در آخر این ASN.1 object جدید، با استفاده از قوانین کدگذاری ASN1 مرتب می‌شود و پس از آن با استفاده از سرویس انتقال به موجودیت پروتکل مورد نظر ارسال می‌شود.

به طور مشابه عملیات سطح بالا در موجودیت پروتکل دریافت کننده‌ی پیام شامل موارد زیر است:

1. datagram ورودی تجزیه می‌شود تا ASN.1 object متعلق به ASN.1 message object استخراج شود. اگر تجزیه با خطا مواجه شود، datagram را نادیده می‌گیرد و عمل دیگری بر روی آن انجام نمی‌دهد.

2. سپس شماره‌ی ورژن پیام SNMP را چک می‌کند. اگر اشتباه باشد، datagram را دور می‌اندازد و عمل دیگری بر آن انجام نمی-دهد.

3. پس از آن موجودیت پروتکل نام گروه داده‌ای که در ASN.1 message object یافته است و همچنین آدرس انتقال مبدا و مقصد datagram را به سرویس احراز هویت مورد نظر ، ارسال می‌کند. که این سرویس ASN.1 object دیگری برمی‌گرداند و یا سیگنال خطا در احراز هویت تولید می‌کند. در موارد دیگر این خطا را با تولید trap و دور انداختن datagram اطلاع می‌دهد و عمل دیگری بر روی آن صورت نمی‌گیرد.

4. در پایان موجودیت پروتکل ASN.1 objectای را که سرویس احراز هویت برگردانده است، تجزیه می‌کند تا ASN.1 object متعلق به PDU به دست آید. اگر تجزیه با خطا مواجه شود data gram دور انداخته می‌شود و عمل دیگری بر آن انجام نمی‌شود. در غیر این صورت با استفاده از نام گروه SNMP ، profile مناسب انتخاب ، و به دنبال آن PDU مورد پردازش قرار می‌گیرد. اگر به عنوان نتیجه‌ی پردازش، پیغامی برگردانده شود؛ آدرس انتقال مقصد پیامی که به عنوان پاسخ باید ارسال شود برابر آدرس انتقال مبدا پیغام دریافت شده است.

1.1.4 ساختارهای مشترک

قبل از معرفی شش نوع PDU از پروتکل، برخی ساختارهای ASN.1 استفاده شده معرفی می شود:

```
-- request/response information
RequestID ::=
INTEGER
ErrorStatus ::=
INTEGER {

noError(0),
tooBig(1),
noSuchName(2),
badValue(3),
readOnly(4)
genErr(5)
}
ErrorIndex ::=
INTEGER
-- variable bindings

VarBind ::=
SEQUENCE {
name
ObjectName,
value
ObjectSyntax
}
VarBindList ::=
SEQUENCE OF
VarBind
```

RequestID ها برای تمییز دادن بین درخواست های برجسته استفاده می شوند. با استفاده از RequestID، یک موجودیت کاربردی SNMP می تواند منابع را با درخواست های برجسته مرتبط کند. در مواردی که سرویس دیتاگرام غیرقابل اطمینان استفاده می شود، RequestID اغلب معانی ساده ای از پیام های شناسایی تکرار شده توسط شبکه را تامین می کند.

نمونه non-zero از ErrorStatus برای اشاره به یک استثنا اتفاق افتاده به هنگام پردازش یک درخواست، استفاده می شود. در این موارد ErrorIndex ممکن است با نشان دادن متغیر در یک لیست مسبب استثنا اطلاعات اضافی فراهم کند.

Variable به یک نمونه از شی مدیریت شده نسبت داده می شود. یک variable binding یا VarBind به جفت "اسم" یک متغیر و "مقدار" آن متغیر ارجاع داده می شود. یک VarBindList یک لیست ساده از اسامی و مقدار متغیرها می باشد. برخی از PDUها تنها به اسم یک متغیر و نه مقدار آن بستگی دارند(مانند GetRequest-PDU). در این مورد، قسمت مقدار binding توسط پروتکل در نظر گرفته نمی شود.

با این حال قسمت مقدار همچنان باید داری syntax و کدگذاری معتبر ASN.1 باشد. بنابراین برای این bindingها بخش مقدار باید دارای مقدار NULL از ASN.1 باشد.

GetRequest-PDU 2.1.4

The form of the GetRequest-PDU is:
GetRequest-PDU ::=
[0]
IMPLICIT SEQUENCE {
request-id
RequestID,
error-status -- always 0
ErrorStatus,
error-index -- always 0
ErrorIndex,
variable-bindings
VarBindList
}

GetRequest-PDU توسط یک موجودیت پروتکل و تنها برای درخواست موجودیت کاربردی SNMP خود تولید می شود. به محض دریافت GetRequest-PDU، موجودیت پروتکل دریافت کننده مطابق با هر قانون قابل اجرا در لیست زیر پاسخ می دهد:

(1) اگر برای هر شی نام گذاری شده در فیلد VarBindها، نام شی نظیر نام شی در دسترس برای عملیات get در دید MIB مربوطه نباشد، آنگاه موجودیت دریافت کننده به مبدا پیام دریافت شده یک GetResponse-PDU با فرمی برابر به جز مقدار فیلد error-status که برابر است با noSuchName و مقدار فیلد error-index که ایندکس مولفه نام شی گفته شده در پیام دریافتی می باشد، ارسال خواهد کرد.

(2) اگر برای هر شی نام گذاری شده در فیلد VarBindها، شی از نوع تجمعی است(مانند چیزی که در SMI تعریف شده)، آنگاه موجودیت دریافت کننده برای مبدا پیام دریافت شده یک GetResponse-PDU با شکل یکسان به جز مقدار فیلد error-status که

noSuchName می باشد و همچنین مقدار فیلد error-index که برابر با ایندکس مولفه نام شی گفته شده در پیام دریافتی می باشد، ارسال می کند.

(3) اگر اندازه GetResponse-PDU تولید شده به عنوان چیزی که توصیف شده از یک حدودی فراتر رود، آنگاه موجودیت دریافت کننده برای مبدا پیام یک GetResponse-PDU به همان شکل ولی با مقدار فیلد error-status برابر با toobig، و مقدار فیلد error-index برابر با zero ارسال می کند.

(4) اگر برای هر شی نام گذاری شده در فیلد VarBindها، مقدار شی را به دلایلی جز موارد بالا بازیابی نشود، آنگاه موجودیت دریافت کننده به مبدا پیام یک GetResponse-PDU به همان شکل ولی با مقدار فیلد error-status برابر با genErr و مقدار فیلد error-index برابر با ایندکس مولفه نام شی گفته شده در پیام دریافتی ارسال می کند.

اگر هیچ یک از قوانین بالا اعمال نشود، انگاه موجودیت دریافت کننده پروتکل برای هر شی نام گذاری شده در فیلد VarBindهای پیام دریافتی، یک GetResponse-PDU مثل همان به مبدا پیام ارسال می کند، مولفه منتاظر GetResponse-PDU نام و مقدار آن متغیر را نشان می دهد. مقدار فیلد error-status برابر noError و مقدار فیلد error-index برابر zero می باشد. مقدار فیلد request-id نیز برابر با همان مقدار فیلد در پیام دریافتی خواهد بود.

The GetNextRequest-PDU 3.1.4

فرم GetNextRequest-PDU مشابه GetRequest-PDU می باشد، به جز نشانه ای(tag) که برای نوع PDU است. در زبان ASN.1 داریم:

```
GetNextRequest-PDU ::=
[1]
IMPLICIT SEQUENCE {
request-id
RequestID,
error-status -- always 0
ErrorStatus,
error-index -- always 0
ErrorIndex,
variable-bindings
VarBindList
}
```

GetNextRequest-PDU توسط یک موجودیت پروتکل و فقط به درخواست موجودیت برنامه SNMP تولید می شود.

پس از دریافت GetNextRequest-PDU، موجودیت پروتکل دریافت با توجه به هر قانون قابل اجرا در لیست زیر پاسخ می دهد:

(1) اگر برای هر نام شی درون فیلد متغیر-مقید(variable-bindings field)، که نام آن به ترتیب دیکشنری(قاموسی) (lexicographically) قبل از نام برخی اشیاء موجود برای عملیات get در دید MIB مربوطه نباشد، پس موجودیت دریافت-کننده به سازنده ی پیام دریافتی، پیامی مشابه GetResponse-PDU می فرستد، جز اینکه ارزش فیلد وضعیت خطا(error-status) noSuchName است، و ارزش فیلد شاخص خطا(error-index) شاخص مولفه نام شی ذکر شده در پیام دریافتی است.

(2) اگر اندازه ی GetResponse-PDU تولید شده به شرح زیر از محدودیت های محلی تجاوز کند، سپس موجودیت دریافت کننده به سازنده پیام دریافتی پیامی مشابه GetResponse-PDU می فرستد، جزاینکه ارزش فیلد وضعیت خطا tooBig است و ارزش فیلد شاخص خطا صفر است.

(3) اگر برای هر نام شی در فیلد متغیر-مقید، ارزش نام شی بعدی در ترتیب قاموسی نتواند به دلایلی توسط قوانین فوق پوشش داده-شده و بازیابی شود، موجودیت دریافت کننده به سازنده پیام دریافتی پیامی مشابه GetResponse-PDU می فرستد، جز اینکه ارزش فیلد وضعیت خطا genErr و ارزش فیلد شاخص خطا، شاخص مولفه نام شی ذکر شده در پیام دریافتی است.

اگر هیچ یک از قوانین فوق اعمال نشود، پس از آن موجودیت پروتکل دریافت کننده به سازنده پیام دریافتی پیام GetResponse-PDU را چنین می فرستد که، برای هر نام در فیلد متغیر-مقید(variable-bindings field) پیام دریافت شده، جزء(مولفه) متناظر در پیام GetResponse-PDU نام و مقدار آن شی ای را نشان می دهد که نامش در ترتیب قاموسی نام تمام اشیاء موجود برای انجام عملیات get از دید MIB مربوطه است، همراه با مقدار فیلد نام جزء داده شده، بلافاصله بعدی آن ارزش. ارزش فیلد وضعیت خطای پیام GetResponse-PDU، noError است و ارزش فیلد شاخص خطا(errorindex) صفر است. ارزش فیلد شناسه درخواست پیام GetResponse-PDU همان مقدار از پیام دریافتی است.

1.3.1.4 مثالی پیمایش جدول (Table Traversal)

یکی از موارد مهم استفاده از GetNextRequest-PDU پیمایش جداول مفهومی اطلاعات درون MIB است. معناشناسی این نوع پیام SNMP، همراه با مکانیسم های مختص پروتکل برای شناسایی نمونه های منحصر بفرد انواع شی در MIB، موجب دسترسی به اشیاء مرتبط در MIB می شود، اگر آنها از یک سازمان جدولی برخوردار باشند.

توسط تبادل SNMP مطرح شده در زیر، یک موجودیت برنامه SNMP ممکن است آدرس مقصد و دروازه گام بعدی برای هر مدخل(entry) یک عنصر شبکه خاص در جدول مسیریابی را استخراج کند. فرض کنید که این جدول مسیریابی دارای سه مدخل زیر باشد:

Destination	NextHop	Metric
10.0.0.99	89.1.1.42	5
9.1.2.3	99.0.0.3	3
10.0.0.51	89.1.1.42	5

ایستگاه مدیریتی به عامل SNMP یک پیام GetNextRequest-PDU حاوی ارزش شناسه شی(OBJECT IDENTIFIER) نشان داده شده به عنوان نام متغیرهای مورد درخواست را ارسال می کند :

GetNextRequest (ipRouteDest, ipRouteNextHop, ipRouteMetric1)

عامل SNMP با یک پیام GetResponse-PDU پاسخ می دهد :

GetResponse ((ipRouteDest.9.1.2.3 = "9.1.2.3"),
(ipRouteNextHop.9.1.2.3 = "99.0.0.3"),
(ipRouteMetric1.9.1.2.3 = 3))

ایستگاه مدیریتی ادامه می دهد :

GetNextRequest (ipRouteDest.9.1.2.3,
ipRouteNextHop.9.1.2.3,
ipRouteMetric1.9.1.2.3))

عامل SNMP پاسخ می دهد :

GetResponse ((ipRouteDest.10.0.0.51 = "10.0.0.51"),
(ipRouteNextHop.10.0.0.51 = "89.1.1.42"),
(ipRouteMetric1.10.0.0.51 = 5))

ایستگاه مدیریتی ادامه می دهد :

GetNextRequest (ipRouteDest.10.0.0.51,
ipRouteNextHop.10.0.0.51,
ipRouteMetric1.10.0.0.51)

عامل SNMP پاسخ می دهد :

GetResponse ((ipRouteDest.10.0.0.99 = "10.0.0.99"),
(ipRouteNextHop.10.0.0.99 = "89.1.1.42"),
(ipRouteMetric1.10.0.0.99 = 5))

ایستگاه مدیریتی ادامه می دهد :

GetNextRequest (ipRouteDest.10.0.0.99,
ipRouteNextHop.10.0.0.99,
ipRouteMetric1.10.0.0.99)

ازآنجا که مدخل دیگری در جدول وجود ندارد، عامل SNMP اشیایی که به ترتیب قاموسی بعد از نام شی شناخته شده وجود دارند را برمی گرداند. این پاسخ پایان جدول مسیریابی را به ایستگاه مدیریتی اعلام می کند.

The GetResponse-PDU 4.1.4

ساختار پیام GetResponse-PDU مشابه GetRequest-PDU می باشد به جزنشانه ای که برای نوع PDU است. در زبان ASN.1 داریم:

```
GetResponse-PDU ::=
[2]
IMPLICIT SEQUENCE {
request-id
RequestID,
error-status
ErrorStatus,
error-index
ErrorIndex,
variable-bindings
VarBindList
}
```

پیام GetResponse-PDU توسط موجودیت پروتکل فقط پس از دریافت پیام GetRequest-PDU، GetNextRequest-PDU، و یا SetRequest-PDU که در جاهای دیگر این سند توصیف شده اند، تولید می شود.
پس از دریافت GetResponse-PDU، نهاد پروتکل دریافت کننده محتوای آن را به موجودیت برنامه SNMP ارائه می دهد.

4. 1. 5 SetRequest-PDU

فرم SetRequest-PDU یکسان با GetRequest-PDU میباشد بجز برای نشانه نوع PDU. در زبان ASN.1 :

```
SetRequest-PDU ::=
[3]
IMPLICIT SEQUENCE {
request-id
RequestID,
error-status -- always 0
ErrorStatus,
error-index -- always 0
ErrorIndex,
variable-bindings
VarBindList
}
```

SetRequest-PDU توسط یک موجودیت پروتکل، تنها با درخواست موجودیت برنامه SNMP ایجاد میشود.

به محض دریافت SetRequest-PDU، دریافت کننده براساس هرگونه قانون مناسب در لیست زیر پاسخ میدهد:

1- اگر برای هر شئ نام گذاری شده در حوزه اتصالات متغیر (variable-bindings field)، شئ برای عملیات SET ، از دیدگاه مرتبط MIB در دسترس نباشد، بنابراین دریافت کننده GetResponse-PDU را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به noSuchName مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با مولفه ی نام شئ در پیام دریافتی میباشد.

2- اگر برای هر شئ نام گذاری شده در حوزه اتصالات متغیر (variable-bindings field)، محتویات رشته مقدار، بر اساس زبان ASN.1 ، نوعی، طول یا مقداری که با چیزی که متغیر نیاز دارد سازگار باشد را مشخص نکند، آنگاه موجودیت دریافت کننده GetResponse-PDU را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به badValue مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با مولفه ی نام شئ در پیام دریافتی میباشد.

3- اندازه ی پیام نوع Get Response، ایجاد شده به گونه ای که در زیر توصیف شده، از یک محدودیت محلی تجاوز میکند، آنگاه موجودیت دریافت کننده GetResponse-PDU را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به tooBig مقدار میگیرد، و مقدار رشته ایندکس خطا، صفر میباشد.

4- اگر برای هر شئ نام گذاری شده در حوزه اتصالات متغیر (variable-bindings field)، مقدار شئ نام گذاری شده به دلایلی که توسط هیچکدام از قوانین ذکرشده پوشش داده نشده، نتواند تغییر کند، آنگاه موجودیت دریافت کننده GetResponse-PDU را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا به genErr مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با مولفه ی نام شئ در پیام دریافتی میباشد.

اگر هیچکدام از قوانین ذکر شده را شامل نشد، آنگاه برای هر شئ نام گذاری شده در حوزه اتصالات متغیر (variable-bindings field) پیام دریافت شده، مقدار متناظر به متغیر اختصاص داده میشود. هر تخصیص متغیر تعیین شده توسط SetRequest-PDU باید تحت تاثیر قرار بگیرد چنانکه به طور همزمان با توجه به تمام تخصیص های دیگر در آن پیام مقدار گرفته است.

موجودیت دریافت کننده آنگاه GetResponse-PDU را با فرم یکسان، به عامل مبداء پیام دریافت شده ارسال میکند، جز اینکه مقدار رشته وضعیت خطا پیام تولید شده به noError مقدار میگیرد، و مقدار رشته ایندکس خطا، برابر با صفر میباشد.

4. 1. 6 Trap-PDU

فرم Trap-PDU به صورت زیر میباشد:

```
Trap-PDU ::=
[4]
IMPLICIT SEQUENCE {
enterprise -- type of object generating
-- trap, see sysObjectID in [5]
OBJECT IDENTIFIER,
agent-addr -- address of object generating
NetworkAddress, -- trap
generic-trap -- generic trap type
INTEGER {
coldStart(0),
warmStart(1),
linkDown(2),
```

```
linkUp(3),
authenticationFailure(4),
egpNeighborLoss(5),
enterpriseSpecific(6)
},
specific-trap -- specific code, present even
INTEGER, -- if generic-trap is not
-- enterpriseSpecific
time-stamp -- time elapsed between the last
TimeTicks, -- (re)initialization of the network
-- entity and the generation of the
trap
variable-bindings -- "interesting" information
VarBindList
}
```

Trap-PDU توسط یک موجودیت پروتکل، تنها با درخواست موجودیت کاربردی SNMP ایجاد میشود. وسیله ای که توسط آن موجودیت کاربردی SNMP آدرس مقصد های موجودیت های برنامه کاربردی SNMP را انتخاب میکند، مبتنی بر پیاده سازی میباشد.

به محض دریافت Trap-PDU موجودیت پروتکل دریافتی ، محتویات خود را به موجودیت کاربردی SNMP ارائه میکند.

اهمیت مولفه اتصالات متغیر Trap-PDU ، مبتنی بر پیاده سازی میباشد.

تفسیر مقدار رشته generic-trap به صورت زیر میباشد:

coldStart Trap 1 .6 .1 .4

تله coldStart(0) دلالت دارد که موجودیت پروتکل در حال ارسال در حال مقدار دهی مجدد خودش میباشد به طوری که تنظیمات عامل یا پیاده سازی موجودیت پروتکل ممکن است تغییر کند.

warmStart Trap 2 .6 .1 .4

تله warmStart(1) دلالت دارد که موجودیت پروتکل در حال ارسال در حال مقدار دهی مجدد خودش میباشد به طوری که هیچ یک از تنظیمات عامل یا پیاده سازی موجودیت پروتکل تغییر نمیکند.

linkDown Trap 3 .6 .1 .4

تله linkDown(2) دلالت دارد که موجودیت پروتکل در حال ارسال یک نارسائی را در یکی از لینک های ارتباطی ارائه شده در تنظیمات عامل تشخیص میدهد.

Trap-PDU از نوع linkDown ، اولین عنصر متغیر اتصالاتش محتوی نام و مقدار نمونه ifIndex برای رابط متاثر میباشد.

linkUp Trap 4 .6 .1 .4

تله linkUp(3) دلالت دارد که موجودیت پروتکل در حال ارسال تشخیص میدهد که یکی از لینک های ارتباطی ارائه شده در تنظیمات عامل درخواست ارتباط دارد (has come up)

Trap-PDU از نوع linkUp، اولین عنصر متغیر اتصالاتش محتوی نام و مقدار نمونه ifIndex برای رابط متاثر میباشد.

authenticationFailure Trap 5 .6 .1 .4

تله authenticationFailure(4) ، دلالت دارد که موجودیت پروتکل در حال ارسال مخاطب یک پیام پروتکلی است که به طور مناسب تصدیق نشده است. در حالی که پیاده سازی های SNMP باید قادر باشند این تله را ایجاد کنند، آنها همچنین باید قادر باشند که انتشار چنین تله هایی را به وسیله یک مکانیزم مبتنی بر پیاده سازی سرکوب کنند.

egpNeighborLoss Trap 6 .6 .1 .4

تله egpNeighborLoss(5) دلالت دارد بر اینکه یک همسایه EGP برای کسی که موجودیت پروتکل در حال ارسال برایش یک جفت EGP (peer) بود، علامت گذاری شده و ارتباط متناظر دیگر بدست نمی آید.

Trap-PDU از نوع egpNeighborLoss، اولین عنصر متغیر اتصالاتش محتوی نام و مقدار نمونه egpNeighAddr برای همسایه تحت تاثیر میباشد.

enterpriseSpecific Trap 7 .6 .1 .4

تله enterpriseSpecific(6) دلالت بر این دارد که موجودیت پروتکل در حال ارسال تشخیص میدهد که تعدادی رویداد مبتنی بر سازمان رخ داده اند.

رشته specific-trap، تله خاصی که اتفاق افتاده را شناسایی میکند.

5. Definitions

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
IMPORTS
ObjectName, ObjectSyntax, NetworkAddress, IpAddress, TimeTicks
FROM RFC1155-SMI;

-- top-level message

Message ::=
SEQUENCE {

version -- version-1 for this RFC

INTEGER {
version-1(0)
},
community -- community name
OCTET STRING,
data -- e.g., PDUs if trivial
ANY -- authentication is being used

}

-- protocol data units

PDUs ::=
CHOICE {
get-request
GetRequest-PDU,
get-next-request
GetNextRequest-PDU,
get-response
GetResponse-PDU,
set-request
SetRequest-PDU,
trap
Trap-PDU
}

-- PDUs

GetRequest-PDU ::=
[0]
IMPLICIT PDU
GetNextRequest-PDU ::=
[1]
```

IMPLICIT PDU
GetResponse-PDU ::=
[2]
IMPLICIT PDU
SetRequest-PDU ::=
[3]
IMPLICIT PDU
PDU ::=
SEQUENCE {
request-id
INTEGER,
error-status -- sometimes ignored

INTEGER {
noError(0),
tooBig(1),
noSuchName(2),
badValue(3),
readOnly(4),
genErr(5)
},
error-index -- sometimes ignored
INTEGER,
variable-bindings -- values are sometimes ignored
VarBindList
}
Trap-PDU ::=
[4]
IMPLICIT SEQUENCE {
enterprise -- type of object generating

-- trap, see sysObjectID in [5]

OBJECT IDENTIFIER,

agent-addr -- address of object generating
NetworkAddress, -- trap
generic-trap -- generic trap type
INTEGER {
coldStart(0),
warmStart(1),
linkDown(2),
linkUp(3),
authenticationFailure(4),
egpNeighborLoss(5),
enterpriseSpecific(6)
},
specific-trap -- specific code, present even
INTEGER, -- if generic-trap is not

-- enterpriseSpecific

```
time-stamp -- time elapsed between the last
TimeTicks, -- (re)initialization of the
network

-- entity and the generation of the

trap
variable-bindings -- "interesting" information

VarBindList
}

-- variable bindings

VarBind ::=
SEQUENCE {
name
ObjectName,
value
ObjectSyntax
}
VarBindList ::=
SEQUENCE OF
VarBind
END
```

6. قدردانی

این یادداشت متاثر از کار گروه توسعه ی SNMP سازمان IETF بود:

Karl Auerbach, Epilogue Technology
K. Ramesh Babu, Excelan
Amatzia Ben-Artzi, 3Com/Bridge
Lawrence Besaw, Hewlett-Packard
Jeffrey D. Case, University of Tennessee at Knoxville
Anthony Chung, Sytek
James Davidson, The Wollongong Group
James R. Davin, MIT Laboratory for Computer Science
Mark S. Fedor, NYSERNet
Phill Gross, The MITRE Corporation
Satish Joshi, ACC
Dan Lynch, Advanced Computing Environments
Keith McCloghrie, The Wollongong Group
Marshall T. Rose, The Wollongong Group (chair)
Greg Satz, cisco
Martin Lee Schoffstall, Rensselaer Polytechnic Institute
Wengyik Yeong, NYSERNet

7. منابع

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, IAB, April 1988.
[2] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1065, TWG, August 1988.
[3] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1066, TWG, August 1988.
[4] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, IAB, August 1989.
[5] Rose, M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, Performance Systems International and Hughes LAN Systems, May 1990.
[6] McCloghrie, K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based Internets", RFC 1156, Hughes LAN Systems and Performance Systems International, May 1990.

[7] Case, J., M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol", Internet Engineering Task Force working note, Network Information Center, SRI International, Menlo Park, California, March 1988.
[8] Davin, J., J. Case, M. Fedor, and M. Schoffstall, "A Simple Gateway Monitoring Protocol", RFC 1028, Proteon, University of Tennessee at Knoxville, Cornell University, and Rensselaer Polytechnic Institute, November 1987.
[9] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.
[10] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)", International

Organization for Standardization, International Standard 8825, December 1987.
[11] Postel, J., "User Datagram Protocol", RFC 768, USC/Information Sciences Institute, November 1980.

Security Considerations
Security issues are not discussed in this memo.

Authors' Addresses
Jeffrey D. Case
SNMP Research

P.O. Box 8593
Knoxville, TN 37996-4800
Phone: (615) 573-1434
Email: case@CS.UTK.EDU
Mark Fedor
Performance Systems International
Rensselaer Technology Park
125 Jordan Road
Troy, NY 12180

Phone: (518) 283-8860
Email: fedor@patton.NYSER.NET
Martin Lee Schoffstall
Performance Systems International
Rensselaer Technology Park
165 Jordan Road
Troy, NY 12180
Phone: (518) 283-8860
Email: schoff@NISC.NYSER.NET