Protocolo SNMP

SNMP (Simple Network Management Protocol) es un protocolo de red estándar utilizado para la administración y monitoreo de dispositivos en una red. Fue desarrollado con el propósito de facilitar la gestión de redes y permite que un administrador o una aplicación recojan datos sobre el estado, rendimiento y posibles problemas de los dispositivos conectados, tales como routers, switches, servidores, impresoras, y más.

SNMP se utiliza principalmente para:

- **Monitoreo de red:** Obtener información sobre el estado de los dispositivos de la red (como uso de CPU, estado de interfaces, rendimiento de la red, etc.).
- **Administración remota:** Administrar remotamente dispositivos, cambiando configuraciones o ajustando parámetros.
- **Detección y notificación de errores:** Los dispositivos SNMP pueden enviar traps o alertas al administrador cuando se detecta un fallo o problema.
- **Recolección de datos estadísticos:** Se pueden recopilar métricas de rendimiento y tráfico de los dispositivos, lo que permite análisis históricos y ajustes a la red para mejorar su eficiencia.

Componentes principales de SNMP

- **Agente SNMP:** Es el software que se ejecuta en el dispositivo administrado (router, switch, servidor, etc.). El agente mantiene información sobre el estado del dispositivo y responde a las solicitudes SNMP enviadas por el manager.
- **Manager SNMP:** Es el software que se ejecuta en la máquina que gestiona la red. Es el encargado de enviar solicitudes al agente SNMP, recopilar información y recibir notificaciones (traps). Los administradores de red utilizan el manager para monitorear y controlar los dispositivos.
- **MIB (Management Information Base):** Es una base de datos jerárquica que define todas las variables (objetos) que pueden ser monitoreadas o configuradas en un dispositivo SNMP. Cada dispositivo tiene su propia MIB que define los parámetros específicos que puede reportar o modificar.
- **OID (Object Identifier):** Es un identificador único para cada variable o dato que puede ser monitoreado en un dispositivo. Los OID representan las métricas o configuraciones específicas que pueden ser consultadas o modificadas mediante SNMP. Cada OID pertenece a un objeto en la MIB y está representado como una secuencia de números.

Operaciones SNMP

- **GET:** El manager solicita información de un agente, pidiendo datos específicos sobre un dispositivo.
- **SET:** El manager envía datos para configurar un parámetro en el dispositivo.
- **GETNEXT:** El manager pide el siguiente valor en la secuencia de la MIB (útil para recorrer la estructura de la MIB).
- **TRAPS:** El agente envía una alerta o notificación al manager cuando se produce un evento específico (fallo de hardware, sobrecarga de CPU, etc.).

Un OID (Object Identifier) es un número único que identifica de manera jerárquica un objeto dentro de la MIB. Cada OID corresponde a una variable que puede ser monitoreada o configurada en un dispositivo SNMP.

Estructura de un OID

Un OID está compuesto por una secuencia de números separados por puntos (.) que describen la ubicación del objeto en la jerarquía de la MIB. Por ejemplo:

1.3.6.1.2.1.1.5.0

Este OID corresponde a la variable que almacena el nombre del sistema en un dispositivo SNMP, en la rama de la MIB llamada system. A continuación, desglosamos este OID:

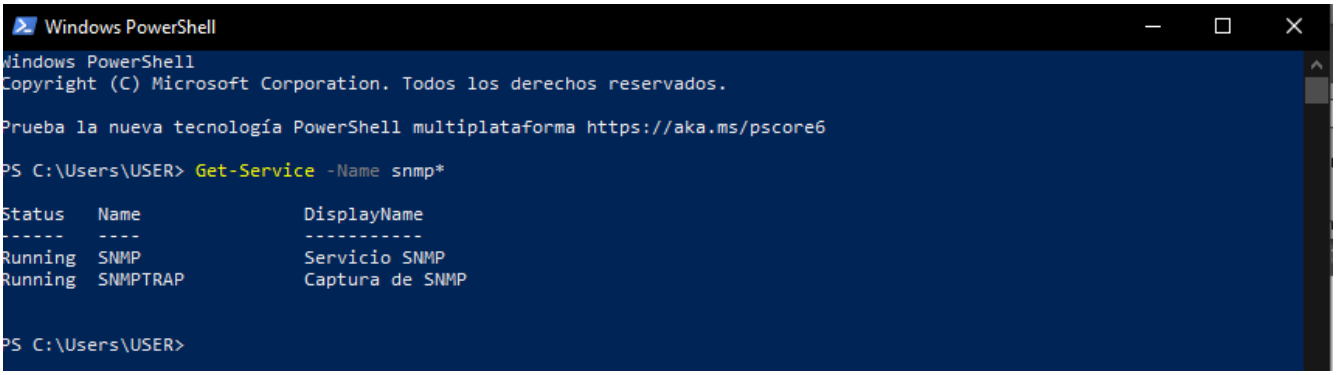
Casos de uso de SNMP

- **Monitoreo del rendimiento de red:** Se pueden usar OIDs para consultar métricas de tráfico en interfaces de red, como la cantidad de bytes enviados o recibidos. Esto es útil para analizar la capacidad de la red y planificar actualizaciones.
- **Monitoreo de la salud de los dispositivos:** Se puede monitorizar el uso de CPU, memoria, temperatura y otros factores clave de dispositivos como routers o servidores.
- **Alertas de eventos críticos:** SNMP permite configurar traps que se disparan cuando se producen eventos críticos, como la caída de una interfaz de red o sobrecarga de recursos.
- **Administración remota:** A través de SNMP SET, los administradores pueden cambiar configuraciones de dispositivos, como modificar direcciones IP, activar o desactivar interfaces, o cambiar contraseñas.

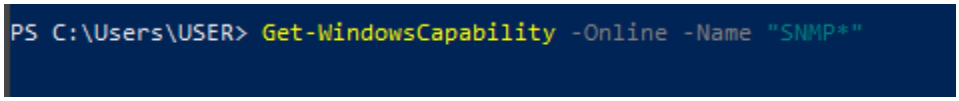
Desarrollo:

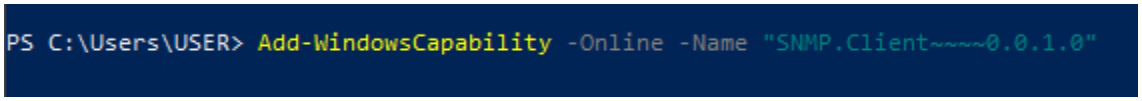
Se va a realizar un programa que obtenga los datos de una PC o varias PC que se encuentren en la comunidad SNMP que se desea para ello utilizaremos los casos de prueba con Ubuntu y con Windows.

Primero se debe de configurar a Windows con el servicio SNMP para ello se hace lo siguiente:



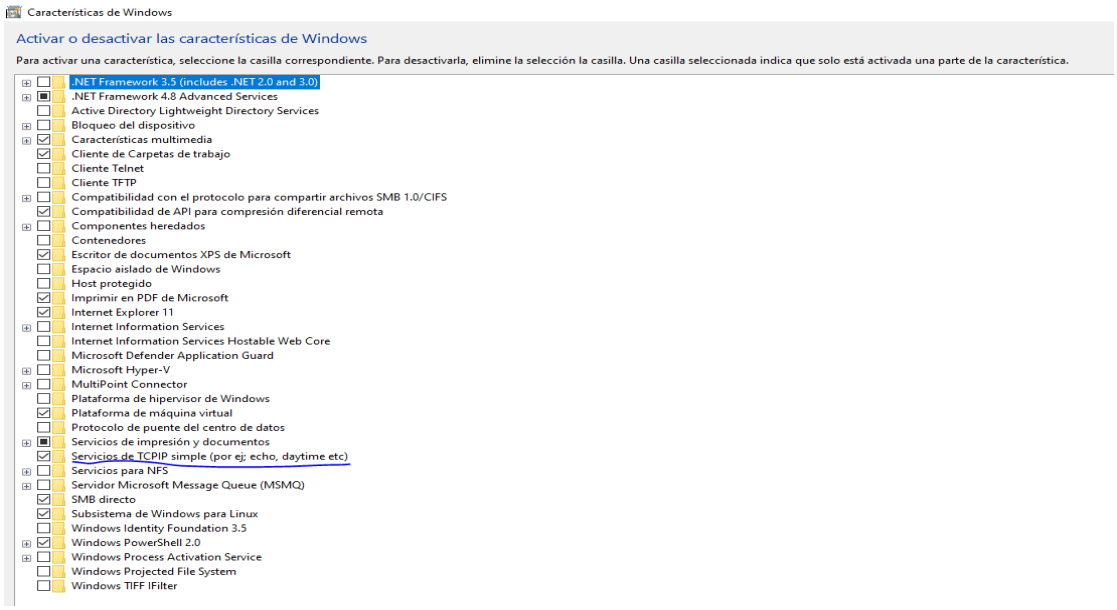
En la imagen se muestra que esta en ejecución el Servicio SNMP y la captura SNMP, en caso de no tener el servicio de SNMP se debe de realizar lo siguiente:



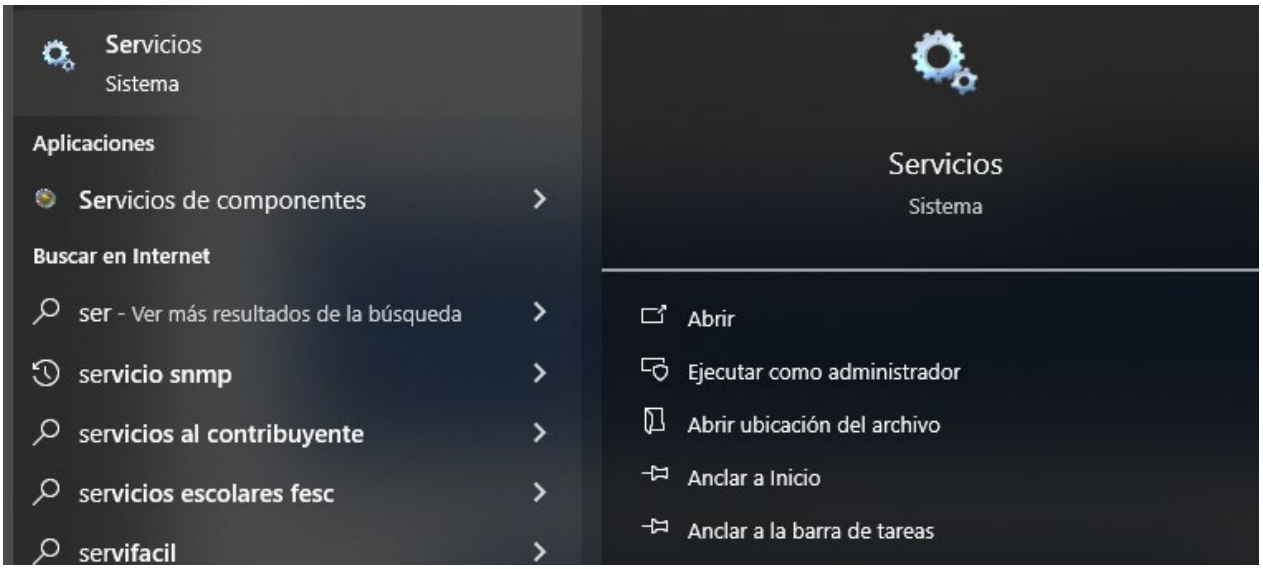


Después se debe de activar la siguiente opción para ello primero se debe de ir a:

Panel de control → Programas → Programas y características → Activar o desactivar las características de Windows.

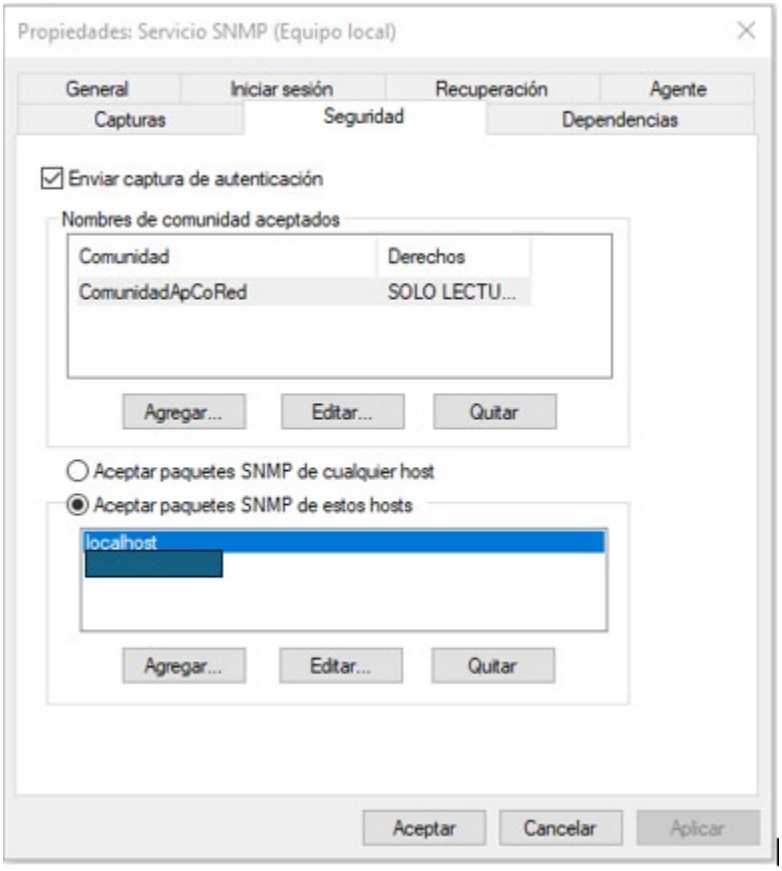


Después debemos ir a servicios:

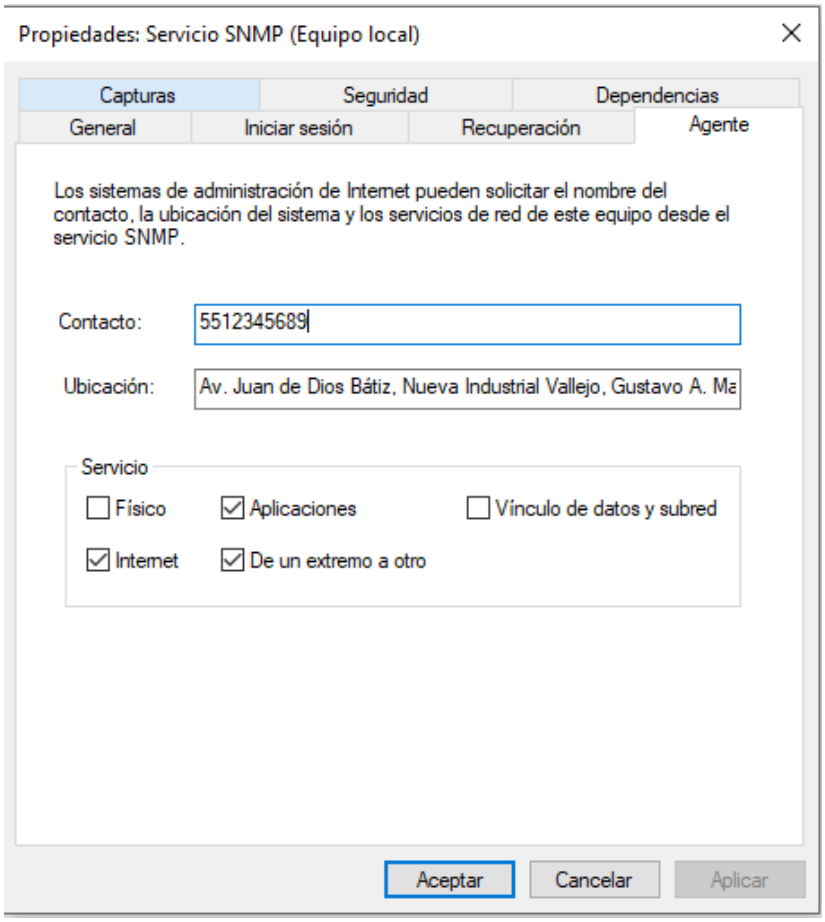


Se debe de configurar el servicio SNMP de la siguiente manera:

Se selecciona el servicio SNMP dando click derecho y seleccionando **Propiedades**, después se debe de ir al apartado de Seguridad, se debe de agregar una comunidad con su respectivo permiso y la IP de quien deseamos recibir paquetes SNMP



Después en la sección de agente se va a agregar un número de contacto y la ubicación, para casos prácticos se utiliza un número aleatorio y la ubicación de la ESCOM.



Para ello se deben de aplicar los cambios efectuados y reiniciar el servicio SNMP.

Antes de establecer la lógica se debe de instalar la siguiente biblioteca con el siguiente comando:

```
sudo apt-get install libsnmp-dev
```

Ahora se procede a establecer la lógica de nuestro código en C como se muestra a continuación:

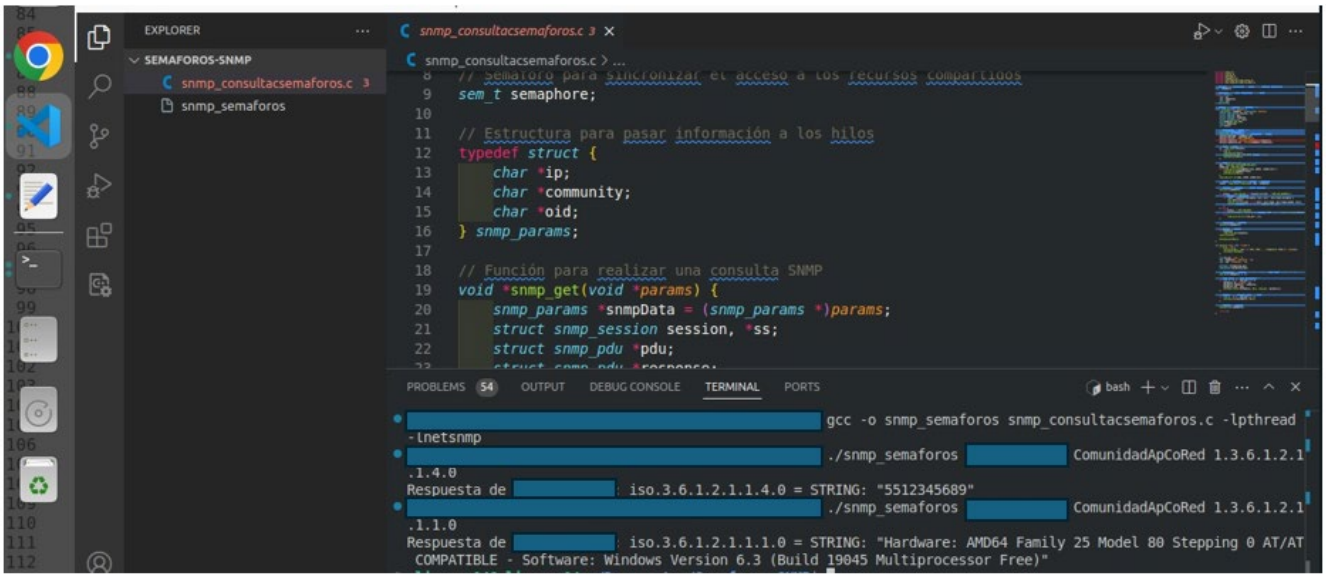
```c
// Semáforo para sincronizar el acceso a los recursos compartidos
sem_t semaphore;

// Estructura para pasar información a los hilos
typedef struct {
    char *ip;
    char *community;
    char *oid;
} snmp_params;

// Función para realizar una consulta SNMP
void *snmp_get(void *params) {
```

```
    snmp_params *snmpData = (snmp_params *)params;
    struct snmp_session session, *ss;
    struct snmp_pdu *pdu;
    struct snmp_pdu *response;
    struct variable_list *vars;
    oid anOID[MAX_OID_LEN];
    size_t anOID_len;
    int status;


    // Abrir la sesión SNMP
    ss = snmp_open(&session);
    if (!ss) {
        snmp_perror("ack");
        snmp_log(LOG_ERR, "UN ERROR OCURRIO!!!\n");
        pthread_exit(NULL);
    }

    // Crear PDU para GET
    pdu = snmp_pdu_create(SNMP_MSG_GET);
    anOID_len = MAX_OID_LEN;
    if (!snmp_parse_oid(snmpData->oid, anOID, &anOID_len)) {
        snmp_perror(snmpData->oid);
        pthread_exit(NULL);
    }
    snmp_add_null_var(pdu, anOID, anOID_len);

    // Enviar la consulta SNMP y obtener la respuesta
    status = snmp_synch_response(ss, pdu, &response);

    // Bloquear el semáforo para acceso exclusivo
    sem_wait(&semaphore);

    if (status == STAT_SUCCESS && response->errstat == SNMP_ERR_NOERROR) {
        // Procesar la respuesta
        for (vars = response->variables; vars; vars = vars->next_variable) {
            char buf[1024];
            snprint_variable(buf, sizeof(buf), vars->name, vars->name_length,
vars);
            printf("Respuesta de %s: %s\n", snmpData->ip, buf);
        }
    } else {
        if (status == STAT_SUCCESS)
            fprintf(stderr, "Error en la consulta SNMP: %s\n",
snmp_errstring(response->errstat));
        else
            snmp_sess_perror("snmp_get", ss);
    }

    // Desbloquear el semáforo
    sem_post(&semaphore);

    // Liberar la memoria
    if (response)
        snmp_free_pdu(response);
    snmp_close(ss);
```

```
    pthread_exit(NULL);
}

int main(int argc, char **argv) {
    if (argc < 4) {
        fprintf(stderr, "Uso: %s <IP1> <IP2> ... <community> <OID>\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    int num_ips = argc - 3;
    char *community = argv[argc - 2];
    char *oid = argv[argc - 1];

    pthread_t threads[num_ips];
    snmp_params params[num_ips];

    // Inicializar el semáforo con valor 1 (solo un hilo puede acceder a la vez)
    sem_init(&semaphore, 0, 1);

    // Crear hilos para cada dispositivo IP
    for (int i = 0; i < num_ips; i++) {
        params[i].ip = argv[i + 1];
        params[i].community = community;
        params[i].oid = oid;
        pthread_create(&threads[i], NULL, snmp_get, &params[i]);
    }

    // Esperar a que terminen todos los hilos
    for (int i = 0; i < num_ips; i++) {
        pthread_join(threads[i], NULL);
    }

    // Destruir el semáforo
    sem_destroy(&semaphore);

    return 0;
}
```

El programa mostrado utiliza semáforos y SNMP para realizar consultas simultáneas a varios dispositivos de red. Los hilos se emplean para gestionar la consulta a cada dispositivo, y los semáforos sincronizan el acceso a la sección crítica donde se imprimen los resultados, evitando conflictos entre hilos. El programa inicia creando una sesión SNMP para cada dispositivo, configurando la versión SNMPv2c y la community (contraseña). Se crea una PDU de tipo GET con un OID que se desea consultar. Luego, mediante la función snmp_synch_response, el programa envía la solicitud y espera la respuesta del dispositivo. Si la consulta es exitosa, se recorren las variables de la respuesta y se muestran los resultados. En caso de error, se maneja a través de funciones SNMP dedicadas. Finalmente, los recursos y la sesión SNMP se liberan.

Para probar el programa se debe de ejecutar de la siguiente manera:



Actividad:

Obtener los siguientes datos del equipo al que estás haciendo la solicitud:

- Sistema Operativo
- Nombre del equipo
- Ubicación
- Interfaz