局域网 IP 扫描器

这个 Python 脚本扫描本地网络以查找活动的 IP 地址。它使用 `ping` 命令检查主机是否可达，并使用多线程来加快扫描过程。信号量限制并发线程的数量，以避免系统过载。脚本接收网络地址（例如，“192.168.1.0/24”）作为输入，并打印网络中每个 IP 地址的运行状态（在线或离线）。

```
import subprocess
import ipaddress
import threading
import os

MAX_THREADS = 255  # 最大线程数

def is_host_up(host):
    """
    使用 ping 命令检查主机是否在线。
    如果主机在线，则返回 True，否则返回 False。
    """
    try:
        # -c 1: 只发送 1 个数据包
        # -W 1: 等待 1 秒钟以获取响应
        subprocess.check_output(["ping", "-c", "1", "-W", "1", host], timeout=1)
        return True
    except subprocess.CalledProcessError:
        return False
    except subprocess.TimeoutExpired:
        return False

def scan_ip(ip_str):
    """
    扫描单个 IP 地址并打印其状态。
    """
    if is_host_up(ip_str):
        print(f"{ip_str} is up")
    else:
        print(f"{ip_str} is down")

def scan_network(network):
    """
    使用线程扫描网络中的活动主机，限制并发线程的数量。
```

```
    """
    print(f"Scanning network: {network}")
    threads = []
    semaphore = threading.Semaphore(MAX_THREADS)  # 限制并发线程的数量

    def scan_ip_with_semaphore(ip_str):
        semaphore.acquire()
        try:
            scan_ip(ip_str)
        finally:
            semaphore.release()

    for ip in ipaddress.IPv4Network(network):
        ip_str = str(ip)
        thread = threading.Thread(target=scan_ip_with_semaphore, args=(ip_str,))
        threads.append(thread)
        thread.start()

    for thread in threads:
        thread.join()

if __name__ == "__main__":
    network_to_scan = "192.168.1.0/24"  # 将此更改为您的网络
    scan_network(network_to_scan)
```