DNIS: A Middleware for Dynamic Multiple Network Interface Scheduling

Ahmed Saeed, Karim Habak, Mahmoud Fouad, and Moustafa Youssef
{ahmed.saeed, karim.habak, mahmoud.fouad, mayoussef}@nileu.edu.eg
Nile University, Smart Village, Egypt

Abstract

Many of today's mobile devices are equipped with multiple network interfaces that can be used to connect to the Internet, including Ethernet, WiFi, 3G, and Bluetooth. Current operating systems, such as Windows and Linux, typically choose only one of the available network interfaces and assign all the traffic to it, even if more than one is connected to the Internet. This results in an obvious underutilization of the available bandwidth from the multiple interfaces. Manual configuration could be used to assign certain destination IPs to certain network interfaces by manipulation the OS's routing table. However, this requires an experienced user and is not sensitive to the dynamic changes in both the network interfaces and the network applications' requirements.

In this work, we present DNIS, a new networking middleware that utilizes all available network interfaces dynamically by distributing the device's traffic on them. DNIS intercepts network connections from the applications and directs them to the best interface. DNIS is composed of two main components: (1) a parameter estimator that estimates the applications' characteristics and requirements as well as interfaces' properties; (2) a scheduler that uses the estimated parameters to assign different connections to network interfaces. We present an implementation for DNIS for the Windows OS and show its performance for different scheduling algorithms. Our initial results show significant enhancement of the overall device's throughput, up to 15%, increasing resource utilization and enhancing the user's experience.

Keywords: Load balancing, middleware services, scheduling multiple network interfaces.

1 Introduction

Making efficient use of the available resources is an important goal for any operating system. This becomes more important for the resource-constrained mobile devices. In today's Internet driven world, network bandwidth has become one of the most important resources. Many of today's mobile devices contain multiple network interfaces, such as Ethernet, WiFi, 3G, and Bluetooth. Combining the bandwidth from all available independent interfaces increases the network resource utilization and enhances the user's experience. Unfortunately, the dominant operating systems today, such as Windows and Linux, typically allow the user to use only one of the available interfaces, *even if multiple of them are connected to the Internet.* A possible solution to this resource underutilization problem is to manually configure the OS's routing tables to assign certain destination IPs to certain network interfaces. However, this requires an experienced user and is not sensitive to the dynamic changes in both the network interfaces and the network applications' requirements.

In this work, we present DNIS: a middleware for Dynamic multiple Network Interface Scheduling. DNIS works by intercepting connection requests and directing them to different network interfaces. It allows the device to be aware of its available network interfaces, their bandwidth, stability, and cost. In addition, it estimates the applications' needs and automatically allocates the traffic carefully to make the best out of the available network resources. We present an implementation of DNIS over the Windows Vista operating system using the Layered Service Provider (LSP) framework [1]. Our initial results show significant enhancement in the network throughput, up to 15%, compared to the default behavior of the operating system.

2 Architecture

Our DNIS middleware has two subtasks:

- Estimation: Where we estimate the applications' needs and characteristics, as well as the properties of the network interfaces.
- Scheduling: Where we assign different connections to different network interfaces based on the

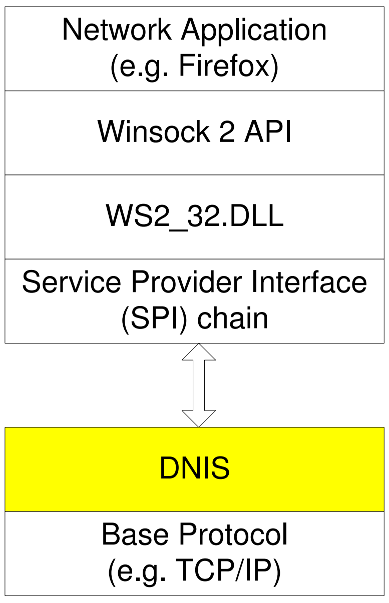

Figure 1: Layered Service Provider Architecture used in Implementing DNIS for the Windows Vista OS.

estimated parameters.

In order to perform these two subtasks, DNIS is composed of two components: (1) A middleware service and (2) a monitoring application.

2.1 Middleware Service

The middleware service is responsible for estimating the different parameters and scheduling the different connections to different network interfaces. It does this by intercepting connection requests from networking applications and rerouting them to the best interface. Similarly, for connection-less protocols, such as UDP, DNIS intercepts send requests and direct them to the best interface.

2.2 Monitoring Application

The monitoring application allows the user to monitor the middleware service as well as specify the different DNIS settings.

3 Implementation

We implemented our DNIS middleware on the Windows Vista OS as a Layered Service Provider (LSP) [1] which is installed in the TCP/IP protocol chain in the Windows operating system (Figure 1).

3.1 Middleware Service Implementation

DNIS uses the same concepts used by firewalls and network proxies to control the network flow. One of its components is a service that is used to intercept socket-based connection-requests and assign proper network interfaces to them. DNIS keeps track of the past behavior of different applications and different interfaces and also takes into consideration the user's preferences to make the decision of assigning connections to certain interfaces. DNIS is based on a windows networking API feature called Service Provider Interface (SPI). SPI defines two different types of service providers: (1) a Transport Service Provider and (2) a Name Space Service Provider

DNIS is implemented as a Transport Service Provider. Data transfer protocols are implemented through a chain that has different layers. The base layer, called the "Base Protocol", e.g. TCP, is responsible for how the protocol works. Other layers, like our service, called "Layered Protocols" are responsible for handling high level traffic control (Figure 1).

Using DNIS we can monitor and keep track of the behavior of different applications and the capacity and stability of different network interfaces.

3.2 Algorithms

In this section, we discuss the different algorithms used for estimation and scheduling in DNIS.

3.2.1 Parameters Estimation

To be able to make the right decision, DNIS needs to estimate the properties of the different available network interfaces as well as the needs and characteristics of the different applications running on the system.

For the network interfaces, DNIS keeps track of the current utilization of the interface, error rate, maximum bandwidth, and buffer size. Other metrics, e.g. [2], are also being investigated.

For the applications, DNIS keeps track of the average number of bytes sent and received per unit time, maximum number of bytes sent and received per unit time, and the application type (realtime (e.g. Skype), browser (e.g. Firefox), unclassified). The application type can be estimated based on the executable name of the process, the ports it uses, and/or its traffic pattern. In addition, the DNIS monitoring application can be used to set this type manually by the user.

3.2.2 Scheduling

We use different algorithms to assign applications to interfaces. The current techniques are:

- Using the default interface: This is the default method used by the Windows OS in which all traffic is assigned to the default network interface.

- Random Selection: in which a new connection is assigned to one of the available network interfaces at random.
- Round Robin: in which we select one of the available network interfaces in a rotating basis.
- Selection Based on the Windows API: in which we select one of the available interfaces using the GetBestInterface() API function that returns the best route to the specified destination IP.
- DNIS Selection Criteria: in which we select one of the available interfaces based on matching the profiles of the applications and the profiles of network interfaces using the estimated parameters.

4 Performance Evaluation

4.1 Setup

To evaluate our middleware, we used two laptops:

1. An HP Laptop (Pavilion Entertainment PC dv4t-1300) with a 54 Mbps wireless card and a 100 Mbps Ethernet card running Windows Vista. Both the wireless and the Ethernet cards were connected to a NetGear wireless router with four Ethernet ports. This machine has DNIS installed and it ran a custom client application for generating the data.
2. A Fujitsu Siemens laptop (AMILO Pro V3405) that has a 100 Mbps Ethernet card connected to the router. This laptop is running Windows XP and a custom server that sends data to the client on the other machine.

To generate the network load, the client starts a number of threads concurrently. Each thread connects to the multithreaded server and opens a connection for 15 seconds. During this period, the server continuously transmits data to the client. Therefore, by controlling the number of threads, we can control the load on the client machine .

4.2 Results

Figure 2 shows the effect of using the DNIS middleware on the performance of the system. The results show that a simple round robin scheduler can achieve up to 15% enhancement in the goodput of the system. For small network loads, the overhead introduced by the DNIS middleware is more than its benefit. However, this overhead quickly decreases as the number of threads increases (at two threads). The middleware can be designed to work in a passive mode, where the scheduler is inactive and the middleware only estimates the system's load. When the system's load exceeds a threshold, the middleware enters an active mode, where the scheduler is activated to achieve the goodput enhancement.

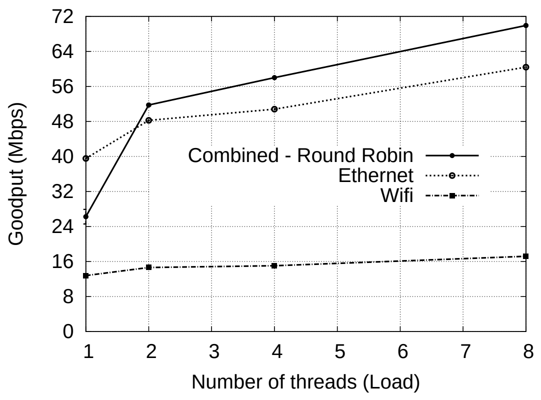

Figure 2: Effect of using the DNIS middleware on the performance of the system using a simple round robin scheduler.

5 Student Research Competition Eligibility

This work is eligible for the student research competition. The first three authors are **undergraduate** students. Their affiliations are included in the title of the proposal.

6 Conclusions

The DNIS middleware allows using multiple network interfaces in parallel. DNIS estimates the applications' and interfaces' parameters and uses this information to schedule different network connections in realtime to the best network interface. Using a simple setup, we showed that DNIS can achieve a 15% enhancement in the network goodput, increasing the newtork utilization and enhancing the user's experience.

References

[1] Wei Hua, Jim Ohlund, and Barry Butterklee, *Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider* , Microsoft Systems Journal, No. 61; Pages 96-113(1999).

[2] Mark Allman, Wesley Eddy, and Shawn Ostermann, *Estimating loss rates with TCP*, SIGMETRICS Perform. Eval. Rev., Vol. 31, No. 3; Pages 12-24(2003).