MobiCom 2009 Poster: DNIS -A Middleware for Dynamic Multiple Network Interfaces Scheduling

Article *in* ACM SIGMOBILE Mobile Computing and Communications Review · April 2010

CITATIONS
0

READS
21

4 authors, including:

Ahmed Saeed
Georgia Institute of Technology
19 PUBLICATIONS **179** CITATIONS
SEE PROFILE

Karim Habak
Georgia Institute of Technology
19 PUBLICATIONS **121** CITATIONS
SEE PROFILE

Moustafa Youssef
Egypt-Japan University of Science and Technol…
219 PUBLICATIONS **5,343** CITATIONS
SEE PROFILE

MobiCom 2009 Poster: DNIS - A Middleware for Dynamic Multiple Network Interfaces Scheduling

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

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. However, 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 under utilization of the available bandwidth. Different bandwidth aggregation techniques suggested altering different layers of the TCP/IP stack which requires applying modifications on the client's stack and/or the cloud, which cannot be widely deployed easily. In this work, we present DNIS, a networking middleware that achieves bandwidth aggregation using per-TCP connection scheduling on different interfaces in a way that is transparent to both the user and the applications. 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 TCP 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 54%, increasing resource utilization and enhancing the user's experience.

I. 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*.

Different approaches has been proposed in literature. Transport layer and network layer approaches require changes to legacy servers [1], require a proxy [2, 3], or are not available to the end user's device directly [4]. Link layer approaches, e.g. IEEE 802.1AX-2008, require homogeneous links, which is not available in mobile environments. Changes of sockets implementation was suggested by [5]. However, this approach requires changes in both the packets' headers and legacy servers.

In this work, we present DNIS: a middleware for Dynamic multiple Network Interfaces Scheduling. DNIS provides a bandwidth aggregation solution that is based on a per TCP-connection scheduling. DNIS tracks applications' and interfaces' behavior. This 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. This is performed transparently from the user and applications. We also present an implementation of DNIS over the Windows Vista operating system using the Layered Service Provider (LSP) framework [6].

II. Architecture

Our DNIS middleware has two subtasks:(1) **Estimation**: Where we estimate the applications' needs and characteristics, as well as the properties of the network interfaces. (2) **Scheduling**: Where we assign different connections to different network interfaces based on the estimated parameters.

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

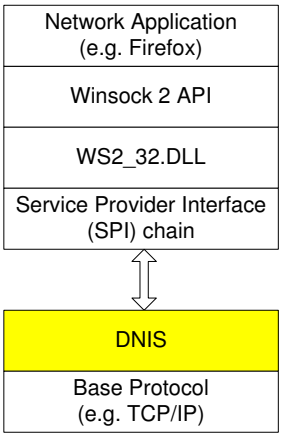

Figure 1: Layered Service Provider Architecture used in Implementing DNIS for the Windows Vista OS.

II.A. 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 directs them to the best interface.

II.B. Monitoring Application

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

III. Algorithms

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

III.A. 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. [7], 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.

III.B. Maximum Throughput Scheduling

For a new connection, the maximum throughput scheduler assigns it to the network interface that will maximize the system throughput. This is equivalent to assigning the new connection to the interface that minimizes the time needed to finish the current system load in addition to the load introduced by this new connection. This algorithm depends on two variables that are computed by the estimation module: (1) Expected time for each interface to finish its current load = Sum of the average load introduced by applications using this interface / average bandwidth calculated for this interface and (2) Expected traffic that the new connection will produce, based on the history of the application creating this connection. We compare the performance of the maximum throughput scheduler to other algorithms in the Section V.

IV. Implementation

We implemented our DNIS middleware on the Windows Vista OS as a Layered Service Provider (LSP) [6] which is installed in the TCP/IP protocol chain in the Windows operating system (Figure 1). 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, e.g. TCP, is responsible for how the protocol works. Other layers, like our service 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.

V. Performance Evaluation

V.A. Setup

To evaluate DNIS, we used two laptops: (1) A Fujitsu Siemens laptop (AMILO Pro V3405) running Windows XP as a server.(2) An HP laptop (X16t) running DNIS on Windows Vista with a custom client for generating traffic. Both machines were connected through a wireless router (SMC7904WBRA2) such that the server connected to the router with a 100Mbps Ethernet link and the client configured to be connected to the router with an 11Mbps wireless link and a 10Mbps Ethernet link to simulate client side bottlenecks, which is usually the case. Each experiment was run for 8 minutes and repeated a number of times to generate the data points. The client establishes new connections with the server according to two Poisson processes representing two classes of applications. The first class represents long-lived connections with a file download size of 150 MByte (rate= λ_{long}), while the second class represents short-lived connections with a file download size of 250 KByte (rate= λ_{short}).

V.B. Results

Figure 2 compares five different schedulers: using only Wi-Fi, using only Ethernet, a round-robin scheduler, a random interface scheduler, and the maximum throughput scheduler. The figure shows that utilizing more than one interface always achieves higher goodput than using one interface. The maximum throughput scheduler, which takes into account the applications' behavior and interfaces' characteristics, achieves higher goodput than other schedulers, which is the sum of the goodput of the two available interfaces. This highlights that DNIS has minimum extra overhead on the protocol stack. When the system load is high, the gap between the different multiple-interface schedulers decreases as the system becomes saturated. The advantage of the maximum throughput scheduler is evident with low to medium loads. The figure shows that the maximum throughput scheduler can perform up to 54% better than using only a single interface.

VI. Ongoing Work

We are currently expanding DNIS in different directions, including designing other schedulers that takes other metrics, such as cost, into account, handling UDP traffic, and applications' characteristics estimation granularity.

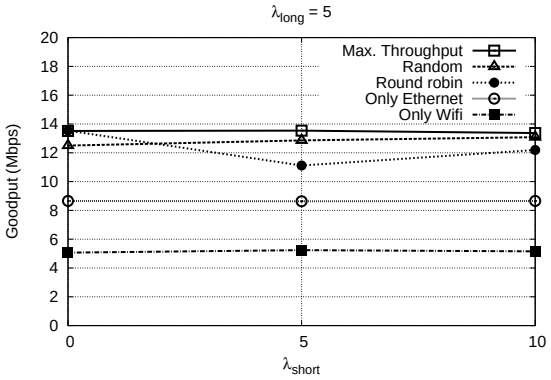

Figure 2: Effect of using the different schedulers on the performance of the system.

References

[1] L. Magalhaes and R. Kravets, *Transport level mechanisms for bandwidth aggregation on mobile hosts*, in: Proc. IEEE ICNP01 (Riverside, Nov 2001).

[2] Kameswari Chebrolu, Bhaskaran Raman, and Ramesh Rao, *A Network Layer Approach to Enable TCP over Multiple Interfaces*, Wireless Networks Volume 11, Issue 5; Pages 637 - 650 (September 2005).

[3] H. Hsieh and R. Sivakumar, *A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts*, in: Proc. ACM MOBICOM02 (Atlanta, Sep. 2002).

[4] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt and S. Banerjee, *MAR: a commuter router infrastructure for the mobile Internet*, 2nd international conference on Mobile systems, applications, and services, Pages: 217 - 230 (2004)

[5] H. Sakakibara, M. Saito and H. Tokuda, *Design and implementation of a socket-level bandwidth aggregation mechanism for wireless networks*, in Proc. of the 2nd annual international workshop on Wireless internet, WICON'06.

[6] 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).

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