PROGRAMMING A LINUX KERNEL MODULE TO CONTROL NETWORK TRAFFIC

University of Pittsburgh

CS 1980 - Capstone Project
Ivan Bondarenko
Madeline Powers
Gennadi Ryan

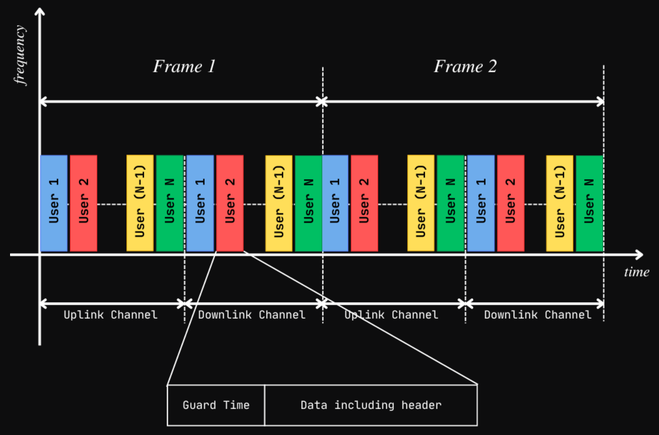
Sponsor: Dr. Luis Oliveira

INTRODUCTION

Our project implements Time Division Multiple Access (TDMA) technology by introducing a dynamic and distributed variant tailored for the Linux Kernel. By leveraging a custom Kernel Module and the *NetController* (netcntlr) userspace program, our system dynamically adjusts network parameters to accommodate changing conditions and user demands. We utilize NETLINK for robust inter-process communication, allowing precise control and real-time adjustments through our TDMA Scheduler. This setup maintains the flexibility of the Linux Kernel while providing advanced network traffic management capabilities directly from the Kernel.

TIME DIVISION MULTIPLE ACCESS (TDMA)

TDMA is a technology used in networking and telecommunications to facilitate the shared use of a single radio frequency channel among multiple users (or hosts) by dividing the signal into different time slots. In our case, we are building a distributed and dynamic version of TDMA. This means that the machines dynamically adjust their own parameters to account for network conditions, new machines joining or dropping out, as well being adjusted manually by a user through our userspace program, *NetController* (netcntlr). The figure below shows TDMA in a single frequency channel dividing up each time slot (Frame) for the N users who need to access the network.

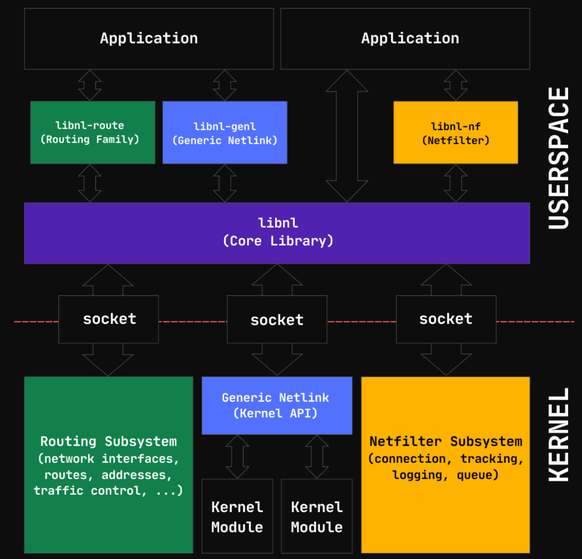


**Time slots contain data with guard period for synchronization*

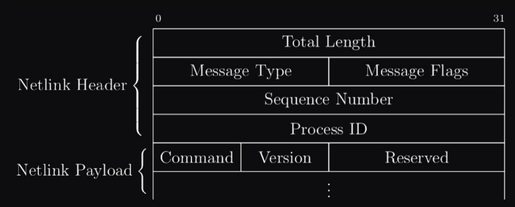
While TDMA is a valuable technique in specific wireless communication and embedded systems contexts, it doesn't make much sense for the Linux Kernel to implement it due to Linux's general-purpose nature, flexibility, and portability design philosophies. As TDMA is generally implemented by higher layers of the network stack by specifically designed hardware and requires more careful and niche mechanisms, it makes sense to instead implement it as an extension to the Kernel as we have done here, through our custom Kernel Module and userspace application...

INTER-PROCESS COMMUNICATION (IPC)

A Kernel is the core component of a computer's operating system, and it acts as the intermediary between the hardware components of the computer and the software it runs. By design, the Kernel is very restrictive to general users of a computer and must be carefully communicated to so as to not disrupt any vital processes or endanger any of the hardware components on the system. Through the use of NETLINK, which is a socket family used for IPC between the Kernel and userspace processes, we can safely and efficiently send messages to the Kernel to interact with the networking hardware on our machines to implement our TDMA Scheduler. By using a high level library, *libnl* as shown in the diagram below, we can pass NETLINK messages to/from the Kernel to our userspace program, *NetController* (netcntlr).



As NETLINK supports multicast, asynchronous, complex, structured data exchange, it was a great solution to achieve our network related communication exchange goals for our TDMA Scheduler. Unlike other IPC approaches such as ioctl or sockets (BSD sockets) that use defined protocols such as TCP to send messages, we manually created our own NETLINK messages and operations, using the header and payload structure as shown in the figure below.



NETCONTROLLER (NETCNTLR)

NetController (netcntlr) is a command-line C program that directly interfaces with our Kernel Module to dynamically change key components of our TDMA Scheduler, thus shaping the underlying Networking implementation of Linux. By using the NETLINK protocols to send messages to our module, we can directly shape core Networking Traffic policies at a high level, using Linux Traffic Control (tc), and key parameters within the Linux Kernel in real-time. From this application, users are able to specify things like offset delays, transmission window widths, time slot width, etc. and can also launch a real-time graphical charting program to see the TDMA Scheduler interacting with the packets being received and transmitted from the system.

LINUX TRAFFIC CONTROL/QUALITY OF SERVICE (TC/QOS)

Linux Quality of Service refers to the Linux Kernel's built-in facilities for scheduling outgoing and incoming network packets. The main unit of QoS is the queueing discipline (QDisc), which is a kernel module/internal component responsible for throttling, or "shaping", network traffic through a variety of strategies.

The core TDMA module is implemented as a custom QDisc, meaning it may be swapped seamlessly with default/built-in QDiscs. This is **one of the first** implementations of TDMA as a core kernel QDisc, rather than as a hardware-bound network device driver. This allows the TDMA QDisc to interoperate with well-known Linux utilities such as iproute2 and tc, allowing experienced network administrators and researchers to configure, experiment with, and modify the QDisc without making modifications to the kernel itself.

Since our use case (dynamic TDMA) requires real-time updates to the QDisc parameters, tc alone is not an ideal mode for interacting with the QDisc. Nonetheless, many modern use cases call only for static TDMA, in which case tc is sufficient. Moreover, future work in this space may show that dynamic TDMA can be achieved through packet classification (e.g. via netfilter), an existing and widely supported feature of the Linux QoS stack.