CS 3410, Project 4 Design Document

Luis Gustavo de Medeiros (lsd53), Ajay Gandhi (aag255)

May 6, 2015

Introduction

This document explains the thought-process and reasoning behind the design decisions taken for the project. It takes an in-depth view at implementation plans for the project and covers design choices (e.g. interrupts vs polling).

Packet Handling Approach

We decided to use a polling system to handle the packets. This approach requires extra communication between cores, which will be covered later in the document.

In addition to polling, we took advantage of the information learned during PA1 and PA2 to create a pipelined system for handling the packets. The first core receives the packets and writes them to a synchronized ring buffer. The second core hashes the packet, classifying it and writing this new information to a second synchronized buffer. The third core reads the categorized packets and updates hashtables corresponding to each datatype. The last core also handles command and print packets accordingly.

Why Polling?

Polling has several advantages over simple interrupts. The most important of these advantages is the ability to handle a large burst of packets. In an interrupt-based system, a core handles packets as soon as they come; that is, as soon as a packet arrives, the core attempts to hash it, classify it, and update corresponding statistics. This entire process is time-consuming, so during a large influx, the majority of packets would be dropped.

Additionally, the polling approach makes it more straightforward to take advantage of multiple cores, since packets are not necessarily handled immediately when they are received.

Datastructures

Two main datastructures will be used to implement the honeypot.

The first datastructure is a simple hashtable, almost identical to that created in Homework 2. There will be multiple instances of the hashtable; each will be used to keep track of a its own type of packet. For instance, one hashtable will be dedicated to keeping track of evil packets and their hashes.

The second datastructure is a synchronized ring buffer. This datastructure is used to keep track of packets between cores. Because the cores act as stages in a pipeline, the structure holding data between the pipeline must be synchronized. Additionally, in contrast to the MIPS processor, these pipelines do act on the same clock cycle. Because packets can arrive at any time and in any volume, they must be queued into a datastructure capable of handling this. Therefore, the synchronized ring buffer can both carry large amounts of packets and maintain synchronization between stages.