6.826 Principles of Computer Systems
Sample Projects

March 11, 2000

1 Air-Traffic Control

In an alternate universe, computer technology developed much faster than aviation technology. As you were walking down Vassar street late one night, a physics experiment in an adjacent building created a time-space singularity that transported you into this alternate universe. When you get there, you are surprised to learn that airplanes are considered to be an extremely dangerous form of travel, because the planes are always crashing into each other on the runways when they take off and land.

You decide to support yourself by building an air-traffic control system that manages the arrivals and departures of airplanes to minimize the crashes. You sell the the Grand Poo-Bah of the country on the idea by promising that there will be no more crashes. Now you have to deliver on your promise (or more precisely, keep convincing the Grand Poo-Bah that a working system is just around the corner if he keeps giving you more money).

Luckily, you took 6.826 the semester before your fateful trip down Vassar street, so you know to do a design of the system in Spec before you start to hack. You should first develop a Spec that says what the system should do. Your system will control an individual airport. Planes will come into the system in one of two ways:

1. They request a take off slot to leave the airport.

2. They come into in your controlled airspace and request a landing slot to arrive at the airport.

Airplanes typically take one of several standard routes into and out of the airport, and you need to be sure that there are no collisions on the runways or on the routes in the airspace surrounding the airport.

You will probably need to model the runways, the airspace, the arrival of the airplanes into the system, and the departure of airplanes from the system. You then develop an implementation that schedules the arrivals and departures. You can make some reasonable assumptions about the arrival rates as long as these assumptions are made explicit. You should strive to let planes take off as soon as possible after they enter the system to depart the airport, and let planes land as soon as possible after they enter the system to land at an airport. Also be sure that airplanes can land before they run out of fuel.

2 White Pages Lookup System

You have recently acquired a tape of all the telephone directories in the United States. The data in this tape was acquired by obtaining a White Pages telephone book from each region in the United States, then hiring cheap labor to type each book into a database.

You decide to start a company to make this data available on the Internet. The spec for your system is simple: given a name, find the number. Or, given a number, find the name. Maybe you even want to deliver more value and let people search by name and state, city, zip code or county. Unfortunately, the data is too big to fit into the main memory of your machine, and is even too big to fit into a single disk. You realize you'll have to have at least 5 disks just to hold the raw data. And you can't just stop there. To access the

data fast enough, you'll have to build indexes, and maybe even store the indexes on extra disks. If you are really ambitious you'll figure out how to keep the data available even if one of your disks fails.

Your job is to figure out how to architect the system. You'll have to make reasonable assumptions about the data set size and distribution, then come up with an implementation that works for those assumptions.

3 Flood Management System

The goal of this project is to specify and implement a flood management system. The system consists of a set of rivers, dams, and water level sensors. When it rains upstream, the water flows into the river and the water level rises. The water then flows downstream, raising the water level as it does so. The system should ensure that the water level at each dam never rises above the height of the dam. If it does, the dam will fail, catastrophically flooding the towns downstream.

Each dam has a spillway that can be opened to let more water out of the lake behind the dam in a controlled way. When it rains and the water level rises upstream, your system must anticipate how much water will flow into the lake behind the dam and preemptively open the spillway to ensure that there is enough room in the lake to hold the excess water. One way to do this is to always keep the spillway completely open. But of course this negates the purpose of the dams, which is to build up waterskiing lakes for wealthy suburbanites. So your system should also have a target water level for each dam and attempt to keep the lake at that level whenever possible.

You will have to model the river basin and come up with a specification that captures the requirements of this system. We are not expecting you to do any complicated differential equations or continuous mathematics; you can use a very simple model that discretizes the system at the granularity of, say, hours. You can also make some reasonable assumptions about the maximum possible change in water levels that can take place over the space of an hour. You will then develop an implementation that decides when to open each spillway to ensure the absence of catastrophic floods.

4 E-commerce

You are designing a modern business-to-consumer e-commerce system, similar to amazon.com. The basic function of the system is to sell books. To do this, it needs a catalog of available books, a way for the user to search the catalog and order books, a shopping basket that records the user's current order, a way for the user to provide credit card and shipping information and be charged for the order, and a way to ship the order.

There are many extensions: remembering the user's information from one day to the next, keeping track of past and pending orders, handling returns, handling disputed charges, keeping track of inventory and ordering from your distributors, aggregating orders from lots of users and recommending related books based on this information, posting sales ranks, recording book reviews from staff or users, one-click ordering, and expanding to other types of merchandise. There are also implementation issues: scalability and availability. Some of these are essential for your site to stay in business, others may be important to grow your business.

The external interfaces of this system are the web pages displayed to the user, the credit card system, and the wholesale distributors that get you books from the publishers. You don't want to do user interface design, so you need a suitable abstraction of a web page, and you also need suitable abstract interfaces to payment and wholesale systems.

Of course a complete spec and implementation for amazon.com, even at a very high level, will be too complicated. You need to choose a suitable set of functions, figure out the abstract state and specs for these functions, and then work out a high-level implementation without getting bogged down in minor details.

5 MBTA

You are designing a train control system for the T. This system gets information from trains about current positions, breakdowns, etc. It gets inputs from operators, both long-term information about desired schedules, inventory of trains, and configuration of the track, and short-term information about how to respond

to unscheduled problems. It has a model of the train and track system, and it issues instructions to train drivers and to switches and signals. It also tells operators the current state of the system.

Your job is not to do detailed modeling of the dynamics of trains, but to abstract away from this to high-level information about the current and expected state of the system, and figure out from this information how to operate the system safely and reasonably efficiently. For simplicity, you can assume that all trains are the same size, and that you get perfect information about the current position and velocity of each train and state of each signal, switch, and section of track. You can't assume that any of these components always carry out your instructions, because they can break. You should not do detailed user interface design, but only a high-level description of what information operators can get and what actions they can take.