6.976/ESD.937 Quantitative Foundations of Engineering Systems Feb 13-15-21, 2006

Lecture 1-4

Lecturer: D. Shah and S. Mitter *Scribe: J. Laracy and D. Shah*

Architecture: Design Principles

Architecture is an *Art* or *Science* of designing engineering system. It is not an *exact* science but there are well-known general principles or guide-lines that can help in designing better engineering system. These lectures were aimed at explaining these principles mainly covered the following three topics:

1. General system design philosophy: provides broad guide-lines for system design.
2. Modularity: the most general principle for architecture design.
3. Interplay between Theory and Architecture: theory leads to better architecture for certain specific systems.

These were explained in detail using the example of Internet, which will be the running example throughout the course.

1. Design Philosophy

Initially, a system is required to design to fulfill certain requirements. For example, Telephone network was designed to fulfill requirement of real-time long-distance communications. Hence, a natural way to begin thinking about architecture of a system is to start from the essential requirement of the system. That is, the first step should be list all the functional and non-functional requirements of the ideal system that one wishes to design. For example, the primary goal or expected requirement from Internet, when it was designed late 1970s-early 1980s, was the multiplexed operation of multiple, independent, heterogeneous networks.

The next step is to use these requirements to derive the design implications. The requirements naturally put constraints on the type of architecture that can be supported. For example, in the context of Internet, requirement of multiplexed operation of independent networks mean that the architecture can not have any *centralized* operations.

Once such design implications or constraints are derived, the next step is the search of technology that satisfies these constraints and allows one to implemented system with desired requirement.

Certainly there is not any straightforward algorithm for implementing the above three steps, going through series of intelligent system specific guesses with multiple iterations of the above steps can lead to a good architecture design. We note that carrying out the above steps require a lot of system specific knowledge and hence it is impossible to have one design principle for all system architecture. The details of the above steps in the context of the Internet are described in the class-slides.

2. Modularity

The principle of modularity is one of the oldest principle of designing engineering system. It has been used widely in all sorts of architecture. The main idea behind modularity is as follows: divide the overall system into smaller sub-system that can be designed independently of each other such that these sub-system can inter-operate by exchanging appropriate information at the interface to provide the functionality of the overall system. Here are some examples.

1. Objected oriented software architecture: each object in software is a module and the objects interact with each other via appropriate interface to provide overall function of desired software system.

2. Divide-and-conquer algorithms design: the algorithmic question is divided into smaller questions and solved separately. The solutions over smaller questions is merged later to provide solution to the overall question.

3. Protocol-stack in Internet: each Internet application requires certain operations. Each operation may require different implementation depending on the underlying infrastructure, requirements, etc. The protocol-stack allows for the flexibility of implementing system specific protocols while retaining the same overall architecture.

3. Interplay: Theory and Architecture

In the context of specific system architecture design, theory helps in providing useful guidelines. We will consider three examples where theory helps in deciding the architecture, in both negative and positive way.

Internet architecture is not an outcome of general theory but result of evolutionary engineering thinking. However, the broad principles employed in the final outcome provide a general way of thinking about such large complex engineering systems.

Example 1: Digital Communication

The main task is to transmit "messages" reliably over a *noisy communication channel.* Now some useful definitions.

Messages. Let message set be $\mathcal{M} = \{m_1, \dots, m_N\}$. Let $\mu_\mathcal{M}$ be distribution according to which these messages are generated from $\mathcal{M}$.

Compression. Consider the message generation as describe above. Let M be random variable that corresponds to the generated message. One can "compress" these messages by certain encoding which is described as follows. An encoding scheme, say $f : \mathcal{M} \to \{0,1\}^*$ maps each message $m_i \in \mathcal{M}$ into some $0-1$ string of length. This mapping is one-to-one so that by looking at the $0-1$ string one can infer what message *index* was. Now the expected length of the encoded message is defined as

$$\mathbf{E}[f(M)] = \sum_{m_i \in \mathcal{M}} \mu_\mathcal{M}(m_i)|f(m_i)|,$$

where $|f(m_i)|$ is the length of the $0-1$ string. Shannon showed that there exists an encoding f^* such that

$$\mathbf{E}[f^*(M)] = H(M),$$

where $H(M)$ is the entropy of M defined as

$$H(M) = -\sum_{m_i \in \mathcal{M}} \mu_\mathcal{M}(m_i) \log \mu_\mathcal{M}(m_i).$$

Further, Shannon showed that $H(M)$ is the lower bound on the expected length of any such encoding scheme.

Channel. Its a discrete memory-less channel. Let each channel input be alphabet from $\mathcal{X}$. On transmission, let the output of channel be from set $\mathcal{Y}$. The channel is noisy and memory-less, that is, when input symbol $x \in \mathcal{X}$ is transmitted the output random variable Y is such that

$$\Pr[Y = y|x] = p_{xy}.$$

Thus the probability transition matrix $P = [p_{xy}]$ characterizes the channel.

Reliable transmission. For each message $m_i \in \mathcal{M}, 1 \le i \le N$ there is a unique code-word $\mathbf{x}_i \in \mathcal{X}^L$, a vector of input symbols of length L. That is, when message m_i is generated then the code-word $\mathbf{x}_i$ is transmitted over the channel. This translation of message into the channel code-word is called *encoding.*

Let $\mathbf{Y}_i$ is the random output received when $\mathbf{x}_i$ is transmitted over the channel. By channel characteristic,

$$\Pr[\mathbf{Y}_i = (y_1, \ldots, y_L) | \mathbf{x}_i = (x_1, \ldots, x_L)] = \prod_{j=1}^{L} p_{x_j y_j}.$$

The decoder, on receiving the output random variable maps it to the appropriate code-word which it believes was transmitted over the channel. Denote $\mathcal{D}$ as this map. This action of mapping is called *decoding* and map is called decoder.

Naturally, an error is created when decoder maps the received output to the *wrong* input code-word. Formally, the probability of error

$$P_e = \sum_{i=1}^{N} \mu_{\mathcal{M}}(m_i) \Pr[\mathcal{D}(\mathbf{Y}_i) \neq \mathbf{x}_i].$$

Finally, we define the rate of the above setup. For this note that since each code-word is of length L, the channel is used L times to transmit each message. The rate is defined as

$$R = H(M)/L,$$

where $H(M)$ is the entropy of the message random variable M. Shannon defined notion of *capacity* given the channel transition matrix P, denoted by $C(P)$ (see below for precise definition). Intuitively, Shannon showed that if the rate R is more than the capacity $C(P)$ then there is no encoding-decoding scheme that can achieve transmission with probability of error P_e arbitrarily small. On the other hand, if R is less than $C(P)$ then there exists a coding scheme that allows one to achieve the error probability arbitrarily close to 0.

Main Implication of Shannon's work. In essence, the above stated work of Shannon suggests the following architectural implication.

Let message be generated according to whatever distribution, first compress them. After compression, we get set of coded messages in form of $0-1$ strings. Now, look at these coded messages and use them to do encoding and decoding for noisy channel. Once channel decoding is done to obtain the transmitted $0-1$ coded string, map it back to the original message.

Now given this *modular* architecture that does encoding of source messages and encoding-decoding at channel independently achieves the *best possible* performance. Thus Shannon's theory suggests natural modularity in the architecture. Such modularity is ubiquitous in current digital communication architecture.

Remark: It is worth remarking here that the modular digital communication architecture, where first compress the source independent of channel on which it is to be transmitted and come up with channel encoding-decoding schemes independent of the data to be transmitted, would have been in practice irrespective of the Shannon's work due to the ease of its implementability. However, if it were found to be not so good in performance one would have indulged in the quest for better architecture. The result of Shannon suggested that this architecture is optimal and hence its better to concentrate on improving the design of this modular architecture rather than looking for any other architecture.

Definition of Capacity. For completeness now we define the capacity and related terms. First we recall the definition of entropy. Given a random variable Z with distribution ν over some discrete set $\mathcal{Z}$. Then its entropy is

$$H(Z) = -\sum_{z \in \mathcal{Z}} \nu(z) \log \nu(z).$$

The $H(\cdot|\cdot)$ is the conditional entropy. Specifically, for two random variables Z_1, Z_2 taking values over space $\mathcal{Z}_1$ and $\mathcal{Z}_2$ respectively

$$H(Z_1|Z_2) = \sum_{z_2 \in \mathcal{Z}_2} \Pr[Z_2 = z_2] H(Z_1|Z_2 = z_2),$$

where

$$H(Z_1|Z_2 = z_2) = -\sum_{z_1 \in \mathcal{Z}_1} \Pr[Z_1 = z_1|Z_2 = z_2] \log \Pr[Z_1 = z_1|Z_2 = z_2].$$

For the above described discrete memory-less noisy, the capacity $C(P)$ is defined as

$$C(P) = \max_{\mu} I(Y;X),$$

where X is distributed according to distribution μ over $\mathcal{X}$ and Y is the output random variable whose distribution (over $\mathcal{Y}$) is induced when X is transmitted as input over channel according to distribution μ (which is governed by channel transition matrix P). The $I(Y;X)$ is called mutual information, where

$$I(Y;X) = H(Y) - H(Y|X) = H(X) - H(X|Y).$$

Here, $H(\cdot)$ is entropy – next we define it.

System Models

Idea of modeling a system is to explain the observed behavior of the system and *leave out everything else.* Thus modeling of a system concerns only what can be modeled, i.e. observed data. Usually, the best model is the *simplest* possible model that can explain the system behavior and nothing more. The search of such a model is a non-trivial task.

We consider broadly two types of system models: (1) Black-box model and (2) Behavioral model. We will first describe the deterministic models.

Black-box model. Black-box model of a system is described by the mapping that maps inputs to outputs – it does not talk about the specific implementation of the system. Such system description naturally imposes the constraint that there is inherent causality in the system. That is, outputs are caused by inputs. This model, while very useful and general enough, is not *universal* since there are many systems where it is not possible to have causal relation. A simple example of black-box models is the description of a plant that is fed by some raw-product and the output is the complete product where amount of the output depends on the raw-product fed to it.

Behavioral model. Such a model of system is described as follows. Let W be set of all possible *signals* or *values* that a system can take at any time-instance. Let T be the set of time over which system evolves – it can continuous or discrete. Let W^T be the set of all expressions that a system can take over time T. Equivalently, W^T is the set of all traces that a system can take over time T.

For example, if $B \subset W^T$ be linear subspace of W^T. Then it corresponds to the set of all trajectories taken by linear systems. The behavioral model is more detailed than the black-box model and hence more general.

Let $B_1, B_2 \subset W^T$ be behaviors of two independent systems. The interconnection corresponds to the following. Define $C_1 = B_1 \times W^T$ and $C_2 = W^T \times B_2$. Then the interconnected system has behavior $C_1 \cap C_2$.

Modeling Uncertainty. Before proceeding further, we note that so far we have considered deterministic models. However, usually one finds oneself in a situation where uncertainty is part of the system. This requires modeling. Here are few ways to model uncertainty: (1) Modeling uncertainty probabilistically, and (2) Modeling uncertainty by identifying the *space* of uncertain component of the system. Depending on the system knowledge, different uncertainty model become relevant.

Next, we describe examples of the behavioral model where the system is described by means of specifying the relation that system parameters must satisfy.

Behavioral Model for Economics. We will describe the behavioral model of economics. The general equilibrium theory says that in an equilibrium state of a *free economic system*, the supply should be equal to the demand of each product. Intuitively, if an economy is at equilibrium but demand is higher than there is an incentive to produce more and vice versa contradicting the fact that system is at equilibrium.

Now suppose there are n products and let variables denoting their prices are $p_1, \ldots, p_n$. Let $S_i : \mathbf{R}_+^n \to \mathbf{R}_+$ be supply functions and $D_i : \mathbf{R}_+^n \to \mathbf{R}_+$ be demand functions for products $i = 1, \ldots, n$. Here S_i maps price vector $\mathbf{p} = (p_1, \ldots, p_n)$ to the amount of supply and similarly for D_i. In equilibrium this means that the equilibrium price-vector $\mathbf{p}^*$ is such that

$$S_i(\mathbf{p}^*) = D_i(\mathbf{p}^*), \quad \text{for all } i.$$

Behavioral Model in Physics. The black-box models are hard to apply in the context of Physical systems as there is no causal relation between system parameters. The way physicists have overcome this as follows: (1) abstract the system behavior mathematically with appropriate model, (2) establishing *universal* relations that are always satisfied by the system. Such a process of modeling and establishing the universal laws is the result of both experimental and mathematical science.

Behavioral Model for Convolution codes. These are linear codes. Let source symbols be $\{w_1, \ldots, w_k, \ldots\}$, $w_i \in \{0, 1\}$. Given the source symbols, we wish to generate the code-words $\{y_1, \ldots, y_k, \ldots\}$ where $y_i = (y_{i\ell}))_{1 \le \ell \le n}$ is n-dimensional $0-1$ vector. These are transmitted over channel and output produced is $\{z_1, \ldots, z_k, \ldots\}$. Decoder maps these back to $\{\hat{w}_1, \ldots, \hat{w}_k, \ldots\}$.

$$y_k = Cx_{k-1} + dw_k,$$
$$x_k = Ax_{k-1} + bw_k,$$

where

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

$d = (111)$ and $b = (01)$ with certain initial conditions.

The decoding is maximum-likelihood. That is map $Y^N = \arg\max \Pr\left[z^N | y^N\right]$. The channel described (as before) is memory-less, hence

$$Y^N = \arg\max \prod_{i=1}^{N} \Pr\left[z_i | y_i\right].$$

The Y^N is guessed as transmitted code-word and corresponding $W \in \{w_1, \ldots, w_k, \ldots\}$ as transmitted message.

Von Neumann Architecture

The Von Neumann provided architecture for designing computer systems. Specifically, the Figure 1 shows the schematic diagram describing different components of a computer system. The main breakthrough of this architectural thinking was in allowing for instruction set and data to reside in the same memory. This architecture allowed separation of *software* and *hardware*. Thus, leading to unparalleled progress of single computer systems.

Now, there has been a lot of subsequent interest in the design of high-performance parallel computers. We first define it. A *Bulk Synchronous parallel computer* contains the following components.

1. Many Components: Processors and Memory.

2. Router: Routes messages from one component to the other component.

3. Facilities for synchronizing some or all the components at regular time interval of L units.

4. Computation happens in super-steps, where in each super-step task allocation consists of a combination of a local computation setps, message transmissions and message receptions. After L steps, global check is made as to whether the super-step has been completed. If ”yes”, go to the next super-steps. If ”no” allocated more steps so as to allow for completion of the task.

Some remarks. The Router is concerned with the communication while components (processors) are concerned with the computation. In this sense, the computation and communication are separated.

The performance of such a system is mainly affected by the time-units L at which one can do synchronization successfully. The lower bound on L comes from Hardware while the upper bound on L comes from Software. Thus, the architecture or system design inherently requires interaction between software and hardware. This lack of *modularity* in terms of software and hardware, which was present in the Von Neumann architecture, is lacking in this parallel computer architecture. As should be clear from description...

In view of Valiant, the lack of this separation between software and hardware is the main reason for the failure. A way to improve the architecture design is to allow for partial modularity between software and hardware which can lead to more successful architecture of parallel computer.

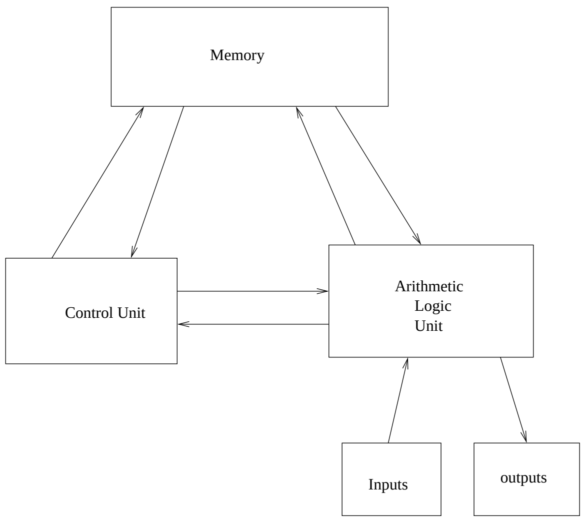

Figure 1: The schematic diagram of architecture of sequential computer system given by Von Neumann.

Definition 1 (Von Neumann bottle-neck) *It is in the separation between CPU and memory, which governs the throughput of the overall system.*

Less primitive way of making big changes in the memory should avoid the bottle-neck between CPU and Memory.

0.1 Some Suggested Reading

- 1977 Turning Lecture on Von Neumann Bottleneck, by Backus.
- On mathematical theory of digital communication, by Shannon.

- Logical design and electronic computing ...., by Von Neumann.

Comments

Related course-slides: http://web.mit.edu/6.976/www/notes/Notes1.pdf