Principals and Practice of Cryptocurrencies

Cornell CS 5437, Spring 2016

Simulation
Example – Gossip

- A message is generated at one machine
- Message processing time: random 0 - 1 sec
- After processing, node chooses random neighbor and sends message
- Sending time: random 0 – 1 sec
Why A Simulation?

- Formal analysis
- Simulation
- Proof of concept experiment
- Working system experiment
Why A Simulation?

<table>
<thead>
<tr>
<th>Formal analysis</th>
<th>Simulation</th>
<th>Proof of concept experiment</th>
<th>Working system experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal Analysis</td>
<td>Simulation</td>
<td>Experiment</td>
<td></td>
</tr>
<tr>
<td>Simple models</td>
<td>Complex models</td>
<td>Real world</td>
<td></td>
</tr>
<tr>
<td>Basic understanding, parameter interaction</td>
<td>Per-run results – cheap runs</td>
<td>Per-run results – expensive runs</td>
<td></td>
</tr>
<tr>
<td>Scale to arbitrary system parameters</td>
<td>Cheap scaling</td>
<td>Expensive scaling</td>
<td></td>
</tr>
</tbody>
</table>

- How complex a model?
  - Over-detailed --> overly complicated
  - Insufficient details --> inaccurate (or wrong)
Event Driven Simulation

**Time-driven simulation**: second by second

**Event-driven simulation**: event by event

Which is more accurate?
**Event Driven Simulation**

**Time-driven simulation**: second by second

**Event-driven simulation**: event by event

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message $m$ arrived at node $i$</td>
</tr>
<tr>
<td>1.3</td>
<td>Message $m$ arrived at node $j$</td>
</tr>
</tbody>
</table>

No tradeoff between efficiency and accuracy
Process-Oriented Simulation

• Use an object per entity
• Store individual object state, and let object react to events

<table>
<thead>
<tr>
<th>Time</th>
<th>Entity</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>node $i$</td>
<td>Message $m$ arrived</td>
</tr>
<tr>
<td>1.3</td>
<td>node $j$</td>
<td>Message $m$ arrived</td>
</tr>
</tbody>
</table>
Input

Includes both **model details** and **runtime inputs**

- Synthetic, e.g.,
  - Topology of huge system
  - Input arrival times
- Traces, measurement-based, e.g.,
  - Input arrival times
  - Processing times
Output

Data collection:
• Output as much data as possible (probably in a log), but not too much – it can easily explode. For example:
  • Message arrival time at each node:
    Can then calculate 90\textsuperscript{th} percentile propagation time without re-running
  • But not every send / processing-start event

Tips:
• Output meaningful data during long runs
• Output execution parameters in log
Executions

Multiple executions
  • Each with different random inputs
  • Warmup
    • Do not consider for statistics
    • E.g., when multiple messages are propagated together, let queues stabilize
  • Or a single long run, divided to sections
Restart and Avoiding It

• Memoize
  • Carefully while you’re debugging...
• Checkpoint
  • For crash handling
  • If you decide continue, to avoid restart
Probability
Probability

• An **experiment** produces a **random results**, $\omega$
• The sample space $\Omega$ is all possible results, $\omega \in \Omega$
• An Event is a subset of the sample space, $E \subset \Omega$
Probability

- Every event $E$ has a **probability** $0 \leq P[E] \leq 1$
- The **conditional probability** of $A$ given $B$ is the probability of $A$ given the $B$ is true; $P[A|B] = \frac{P[A \cap B]}{P[B]}$
- A **random variable** is a function from the sample space to a discrete or continuous range
- A random variable has a **Cumulative Distribution Function** (CDF):
  \[ F_X(x) = P(X \leq x) \]
  \[ F_X(x) \xrightarrow{x \to -\infty} 0, \quad F_X(x) \xrightarrow{x \to \infty} 1 \]
- A discrete random variable has a probability per value
- A continuous random variable has a **probability distribution function**
  \[ f_X(x) = F'_X(x) \]
Probability

- The **mean** of a variable $X$ is
  \[ E[X] = \int_{-\infty}^{\infty} xf_X(x)dx \]

- The **variance** of a variable $X$ is
  \[ \text{var}[X] = E[(X - E[X])^2] = E[X^2] - (E[X])^2 \]
Example – Bernoulli distribution

• Bernoulli distribution:
E.g., due to a biased coin toss, resulting in heads ($\omega_1$) or tails ($\omega_2$)

$$X = \begin{cases} 1 & \text{heads} \\ 0 & \text{tails} \end{cases}$$

$$P[X = 1] = p, P[X = 0] = 1 - p$$

$$E[X] = 1 \cdot p + 0 \cdot (1 - p) = p$$

$$\text{var}[X] = p(1 - p)$$
Example – Normal Distribution

E.g., height, measurement errors

$$N(\mu, \sigma^2)$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$E[X] = \mu$$

$$\text{var}[X] = \sigma^2$$

The standard normal distribution is $N(0, 1)$. 
Example – Normal Distribution

The law of large numbers: The average of Independent and Identically Distributed (IID) random variables converges to the mean of the distribution they are sampled from.

The central limit theorem: the average of IID random variables is approximately normally distributed.
Example – Exponential Distribution

E.g., interval between phone calls

\[ f(x) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

\[ F(x) = \begin{cases} 1 - e^{-\lambda x} & x \geq 0 \\ 0 & \text{otherwise} \end{cases} \]

\[ E[X] = 1/\lambda \]

\[ \text{var}[X] = 1/\lambda^2 \]
Memorylessness

For an exponential random variable, $F(x) = 1 - e^{-\lambda x}$:

$$P[X > s + t|X > t] = \frac{P[X > s + t \cap X > t]}{P[X > t]}$$

$$= \frac{P[X > s + t]}{P[X > t]} = \frac{1 - F(s + t)}{1 - F(t)} = \frac{e^{-\lambda(s+t)}}{e^{-\lambda t}}$$

$$= e^{-\lambda s} = 1 - F(s) = P[X > s]$$

It doesn’t matter how long you have been waiting for the bus
Psuedo Random Number Generator

- No real randomness
- PRNG has state and output
  - State changes on every output request (loops, but usually not an issue)
  - Can be reset with given seed

```python
>>> import random
>>> random.random()
0.28651040107085957
>>> random.random()
0.1796791064694051
>>> random.seed(42)
>>> random.random()
0.6394267984578837
>>> random.seed(42)
>>> random.random()
0.6394267984578837
>>> random.random()
0.02501075522266693
```
Use Object PRNG

- Program modular
- Reset global PRNG in each module?
- No – object PRNG per module
- Seed module’s PRNG on init

```python
>>> randA = random.Random(42)
>>> randB = random.Random(42)
>>> randA.random()
0.6394267984578837
>>> randB.random()
0.6394267984578837
>>> randA.random()
0.025010755222666936
>>> randB.random()
0.025010755222666936
```
PRNG for Simulation

- Different seeds for different runs
- Reproducibility (mostly for debugging)
  - Manually change between runs
  - Seed time, but record seed for reproduction
Inverse Transform Sampling

\[ f(t) = \begin{cases} 
0.5 & 1 \leq t < 3 \\
0 & \text{otherwise}
\end{cases} \]

\[ F(t) = \begin{cases} 
0 & t < 1 \\
0.5(t - 1) & 1 \leq t < 3 \\
1 & t \geq 3
\end{cases} \]

\[ F^{-1}(u) = 1 + 2u \]

\[
\begin{aligned}
\text{>> randA = random.Random()}
\text{>> x = 1 + 2 * randA.random()}
\end{aligned}
\]
Statistical Analysis

Given a finite set of measurements

• Estimate the properties of the sampled space
• Estimate the estimation accuracy

Stop when the accuracy is sufficient.
Statistical Analysis

Take a sample of size $n \{x_i, 1 \leq i \leq n\}$ of independent measurements. E.g., simulation propagation times from $n$ runs.

Sample are taken from a population with probability distribution with mean $\mu$ and variance $\sigma^2$. ($\mu$ and $\sigma$ are unknown)

- The sample mean is:
  \[
  \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
  \]

- The sample variance is:
  \[
  S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
  \]
Statistical Analysis

But the sample mean is a random variable.

The mean of $\bar{x}$ is $\mu$.

So $\bar{x}$ is an estimator of $\mu$

- $\bar{x}$ is not $\mu$, and
- $S$ is not $\sigma$
Statistical Analysis

What is the variance of $\bar{x}$?
Statistical Analysis

What is the variance of $\bar{x}$?

$$\text{var}[\bar{x}] = \frac{\sigma^2}{n}$$

More samples $\Rightarrow$ smaller variance $\Rightarrow$ $\bar{x}$ probably closer to $\mu$

But we can only take a finite number of samples $n$.
And we don’t have $\sigma$, only $S$.
So we need to estimate $\sigma^2$. 
Statistical Analysis

What is the mean of $S^2$?
Statistical Analysis

\[ S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left( x_i - \frac{1}{n} \sum_{j} x_j \right)^2 = \]

\[ = \frac{1}{n-1} \sum_{n=1}^{n} \left( (x_i - \mu) - \frac{1}{n} \sum_{j} (x_j - \mu) \right)^2 = \]

\[ = \frac{1}{n-1} \sum_{i} \left( (x_i - \mu)^2 + \frac{1}{n^2} \left( \sum_{j} (x_j - \mu) \right)^2 - \frac{2}{n} (x_i - \mu) \sum_{j} (x_j - \mu) \right) = \]

\[ = \frac{1}{n-1} \sum_{i} (x_i - \mu)^2 + \frac{1}{n(n-1)} \left( \sum_{j} (x_j - \mu) \right)^2 - \frac{2}{n(n-1)} \sum_{i} (x_i - \mu) \sum_{j} (x_j - \mu) = \]
Statistical Analysis

\[
\frac{1}{n-1} \sum_{i} (x_i - \mu)^2 + \frac{1}{n(n-1)} \left( \sum_{j} (x_j - \mu) \right)^2 - \frac{2}{n(n-1)} \sum_{i} (x_i - \mu) \sum_{j} (x_j - \mu) = \\
= \frac{1}{n-1} \sum_{i} (x_i - \mu)^2 - \frac{1}{n(n-1)} \left( \sum_{i} (x_i - \mu)^2 + \sum_{i} \sum_{j \neq i} (x_i - \mu)(x_j - \mu) \right) = \\
= \frac{1}{n} \sum_{i} (x_i - \mu)^2 - \frac{1}{n(n-1)} \sum_{i} \sum_{j \neq i} (x_i - \mu)(x_j - \mu) \\

E[S^2] = \frac{1}{n} \sum_{i} E[(x_i - \mu)^2] - \frac{1}{n(n-1)} \sum_{i} \sum_{j \neq i} E[(x_i - \mu)(x_j - \mu)] = \sigma^2
\]
Statistical Analysis

What is the mean of $S^2$? $\sigma^2$. 
Confidence Interval

With the standard normal distribution $N(0, 1)$, define $z_{\alpha/2}$ to be the point for which the integral to the right is $\alpha/2$.

(Tables online)

$$P[-z_{\alpha/2} \leq z \leq z_{\alpha/2}] = 1 - \alpha$$

For example, for $\alpha = 0.05$, $z_{0.025} = 1.96$
Confidence

So how accurate is the estimate $\bar{x}$?

If the $x_i$’s are normally distributed: $x_i = N(\mu, \sigma)$, let

$$Z = \frac{\bar{x} - \mu}{\sigma} \sqrt{n}$$

$Z$ is normally distributed $Z = N(0, 1)$. So

$$P\left[-z_{\alpha/2} \leq Z = \frac{\bar{x} - \mu}{\sigma} \sqrt{n} \leq z_{\alpha/2}\right] = 1 - \alpha$$

$$P\left[\bar{x} - \frac{z_{\alpha/2} \sigma}{\sqrt{n}} \leq \mu \leq \bar{x} + \frac{z_{\alpha/2} \sigma}{\sqrt{n}}\right] = 1 - \alpha$$

For confidence level $1 - \alpha$, the confidence interval is

$$\left[\bar{x} - \frac{z_{\alpha/2} \sigma}{\sqrt{n}}, \bar{x} + \frac{z_{\alpha/2} \sigma}{\sqrt{n}}\right]$$
Confidence Interval

- Since we don’t have $\sigma$, we approximate with $S$.
- Although the $x_i$’s are not necessarily normally distributed, according to the central limit theorem, it’s a good approximation for their sum. Thumb rule: $n \geq 30$. 
Example

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = 2.05
\]

\[
S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 = 0.009
\]

\[
z_{0.05/2} = 1.96
\]

\[
\bar{x} \pm z_{0.05/2} \frac{S}{\sqrt{n}} = [1.99, 2.11]
\]
Statistical Analysis

Given a finite set of measurements

• Estimate the properties of the sampled space
• Estimate the estimation accuracy

Stop when the accuracy is sufficient. E.g., the confidence interval is of length 0.01sec with a confidence level of 95%.
Difficulty and Block Interval
Bitcoin’s Block Difficulty

- Hash ($\text{SHA}256^2$) of legal block is smaller than a target
- Target is a 256-bit value
- Stored as a 32-bit field called bits in blocks

Bits:

$$0x180BC409$$

Target:

$$\text{0BC409} \cdot 2^{8(18-3)}$$
Bitcoin’s Block Difficulty

- Hash (SHA256^2) of legal block is smaller than a target
- Target is a 256bit value
- Stored as a 32bit field called bits in blocks
- Largest target (target_{max}) is defined by bits 0x1d00ffff: 0x00000000000000000000000000000000
- Difficulty is defined with respect to the largest target:
  \[ \text{difficulty} = \frac{\text{largest target}}{\text{target}} \]
How long does it take to find a value smaller than
0x0000000000000000000000000000000000000000000000000000000000000000
Or simply:
0x0000000000000000000000000000000000000000000000000000000000000000
?

But wait – the nonce field size...
Bitcoin’s Block Difficulty

How is the target adjusted?

• Once every 2016 blocks (2016/7/24/6 = 2 weeks)
• 2016 blocks ago was $t_0$
• Last block was $t_f$
• Total time is $\Delta = t_f - t_0$ seconds
• Difficulty before $D_{old}$
• Estimate of total hashes calculated: $2016 \times 2^{32}D_{old}$

We want to find $D_{new}$ such that it will take the system 10 minutes on average to find a block. HW
Agenda

• Difficulty calculation
• Time to find a block
• Difficulty automatic tuning
• Minimum of two exponentials and fair mining