**Bancor Simulator**

---

# A Simulator for Market Analysis Under Bancor Protocol

Kenny Kong, Emin Gün Sirer

---

**July, 2017**

*At Cornell University*

## How To Contact With Us:


- **Kenny Kong**
  **Address**: SEIEE Buildings 1-441
  800 Dongchuan Road
  Shanghai Jiao Tong University
  Shanghai, China 200240

  **Email**: klk316980786@sjtu.edu.cn

- **Emin Gün Sirer**
  **Address**: 438 Gates Hall
  Computer Science Dept.
  Cornell University
  Ithaca, NY 14853

  **Email**: egs@systems.cs.cornell.edu


- **Where to find our project?**
  `https://github.com/Ohyoukillkenny/Bancor-Simulator` Bancor-Simulator

# Table of Contents


1 Introduction . . . . . 4
1.1 Quick Scan of Bancor Protocol . . . . . 4
1.2 Introduction of *Bancor Market* . . . . . 6
1.3 Introduction of *Classic Market* . . . . . 7
2 The Framework of Simulator . . . . . 8
2.1 Smart Token . . . . . 8
2.2 Customer . . . . . 9
2.3 Bancor Market . . . . . 10
2.4 Classic Market . . . . . 10
3 Modeling and Operating of Simulator . . . . . 11
3.1 Market Simulating Model . . . . . 11
3.2 Operating Algorithm . . . . . 14
4 The Analysis of Results . . . . . 17
4.1 Indexes for Measuring Market Performance . . . . . 17
4.2 Analysis Based on Price . . . . . 17
4.3 Analysis Based on Transaction . . . . . 19
5 Conclusion and Future Work . . . . . 23

# 1 Introduction

Bancor holds the record for the biggest crowd-funding, ever, in the history of mankind. Admittedly, ... However, ... the robustness and efficiency of Bancor remains to be explored.[2]

Therefore, we build a simulator which could moniter market performance under Bancor protocol, which assists in showing whether Bancor protocol is efficient and safe enough for managing a market with customers coming in and out, launching massive amount of transaction orders, and illustrating whether Bancor offers people an superior platform to deal with business.

After conducting experiments in multiple circumstances and data analyzing as well, results show that Bancor protocol is flawed in three aspects: (1) The problem of "Double Coincidence of Wants" Bancor wants to solve might not exist in real world. Even assuming this problem does happen, Bancor protocol cannot ensure its superiority compared with normal market. (2) The price of smart token, i.e. currency in Bancor protocol could fluctuate significantly, especially when customers generate close valuations of smart token. (3) Bancor protocol cannot fully process multiple transaction orders launched simultaneously, especially when market size is small.

In this document, we try to explain detailed design of this simulator and how this simulator works to simulate the behaviors of both Bancor Market and Classic Market. Also, based on experiment results from simulation, we show our analysis on Bancor protocol.

And this document is organized as follows. In Section 2, we introduce the framework of simulator. In Section 3, we present the simulating model and how to operate our simulator. We give the analysis of simulating results in Section 4. Finally, we draw conclusions in Section 6.

## 1.1 Quick Scan of Bancor Protocol

The key idea of Bancor Protocol is to create a market, i.e., transaction platform between different virtual currencies' pools, by keeping reserve of currencies with a constant ratio. [1] Bancor argues that due to the problem of "Double Coincidence of Wants", it is hard for virtual currencies to make transactions or exchanges between each other. However, after adding a reserve with a constant ratio, customers from different virtual currency pools could directly make deal with the reserve. Therefore, the transaction order can always be satisfied no matter whether there is a real buyer or seller who is launching matched orders at the same time.

In case the reserve between currencies is exhausted, Bancor leverages a mathematical equation for reserve ratio controlling, which is shown as below:

$$P = \frac{B}{S \times \mathrm{CRR}}, \tag{1}$$

where $P$ denotes the price of *smart token*, $B$ corresponds to the balance of *reserve token* market reserve holds, $S$ means the total smart tokens supplied by market and CRR is the constant reserve ratio Bancor protocol wants to maintain.

In fact, there are two kinds of tokens in Bancor protocol. One is the smart token, which refers to the token issued by those who want to set up the market, backing up for the market's reserve. And another is reserve token, which represents the token that has already existed in the world which market builder want to trade with.

In Bancor protocol, once market is established, if buyers try to buy with reserve tokens they hold, the protocol will issue smart tokens to buyers – in other words, the reserve tokens in buyers' hands are converted into smart tokens. In the meanwhile, the price of the smart tokens will increase. For instance, assuming originally one smart token has the same value with one reserve token, now its value might equal with two reserve tokens as customers use reserve tokens to buy smart tokens. While in contrast, when sellers want to sell a certain number of smart tokens to get reserve tokens, Bancor converts/destroys these number of smart tokens to reserve tokens, and decrease the price of smart token. And basically what Eqn.(1) does is to automatically alter the price of smart token under two circumstances we have just mentioned.

More intuitively, what Bancor protocol actually does is to take advantage of price fluctuation to ensure the stability of reserve balance. Imagine there is a market with initial reserve balance, when the reserve balance accumulates (people buy smart tokens consuming reserve tokens), the price of smart tokens will also go high. Therefore, people will hesitate to buy smart tokens since price might exceed their valuation, while eager to sell smart tokens, which decreases the current reserve balance and turns it back to initial amount. Ditto for situation when the reserve balance goes low.

However, as Bancor protocol is built on such an idealized idea with really simple mathematical foundation – actually the Eqn.(1) is the whole idea, it is thoughtless about three conditions and possesses poor robustness.

One thoughtlessness stems from **the frequent price fluctuation advocated by Bancor protocol, which might obstruct transactions in the market**. Technically, after every transaction, the Bancor protocol will adjust the price of smart token to balance the reserve. However, this mechanism might hamper some transaction orders from being finished. For instance, there are two buyers coming into the market who try to buy smart tokens simultaneously. And Bancor protocol should give these two buyers chance to *make valuations* of smart token's price, where making valuation means buyer will determine a maximum purchasing price – if in transaction, the smart token's price is less than maximum purchasing price, the customer will happily to see his order being finished; otherwise he will cancel the transaction. And if customers can not make valuation, it will be unfair, since customers might cost or gain unexpected amount of money to buy or sell smart tokens since the price of smart token is always fluctuating, which might not be exactly what he sees in market. Therefore, go back to the example, these two buyers will propose orders with valuations higher than current price of smart token which we assume is $1. If these two buyers want to buy the smart token with valuation $1.05 and $1.10 respectively in a huge amount, which will lead the current smart token's price to be doubled

to more than $2, there will only one transaction order being finished no matter which order market tries to process first. But why not for market updating the current price information and showing to customers every time when there is a transaction being finished? In fact, there might be a huge number of customers in market trying to make transaction. If market locks the trading entry every time when transaction is generated, the market will suffer with unbearable low throughput capacity and efficiency.

The second flaw of Bancor protocol is that **it neglects the potential abnormal marketing behaviors of customers, which might bleed market's reserve.**[2] Bancor assumes that customers will always follow the law of price, i.e., people are always eager to buy but unwilling to sell when price of the goods (here is the smart token) is low. However, there might be market panic which totally destroys the functioning of price law. For instance, unfounded news about your system might overtake social media. Let's suppose that people got convinced that your CEO has absconded to a remote island with no extradition treaty, that your CFO has been embezzling money, and your CTO was buying drugs from the darknet markets and shipping them to his work address to make a Scarface-like mound of white powder on his desk. Soon, everyone in the market will rush to sell the goods he holds, as he is entirely convinced the goods will be nothing but dust in the near future. Same things happen when there is good news coming in market. When designing Bancor protocol, developers have not fully consider these extreme cases, which might cause fatal crumble of market orders.

Thirdly, Bancor protocol aims to solve **"Double Coincidence of Wants" problem, which actually might not be a problem in real-world market** as no previous study evidence this problem's existence. Further, even assuming the "Double Coincidence of Wants" does exist, it is unsubstantiated to say it will cause damage to market order and customers' benefits as Bancor does not show its superiority compared with normal market laws by solid experimental results.

Therefore, in this document, we build a simulator by constructing a market under Bancor protocol and imitating customers' behaviors to try to explore whether Bancor protocol is prepared to be put on the real-world market. Also, in order to highlight the Bancor protocol's unique features, we simulate the market under classic marketing rules for comparison.

### 1.2 Introduction of *Bancor Market*

The *Bancor market* is the market that is operated under Bancor protocol, processing the transaction orders from customers. Since customers might launch multiple orders simultaneously, we label the market with time, and in every time slot, there might be several customers trying to make transaction orders and putting them on market. Then, the Bancor market will process these orders by Bancor protocol, and correspondingly change the price of goods, i.e., smart tokens. In the end of one time slot, also the beginning of next time slot, the mar-

ket will present the current price to customers, which affords basis for customers who come in in the next time slot making valuation and initiating orders.

### 1.3 Introduction of *Classic Market*

The *Classic market* refers the market which obeys the classic **limited orders?**. In classic market, buyers and sellers also trade with others by lauching transaction orders to the market, and then let market to process their order and feedback result. However, in classic market, buyers can not buy goods directly if there is no selling order which meets buyers' needs, and ditto for sellers who cannot finish orders when no one in the market can satisfy their demands. For instance, if a customer wants to buy 5 rabbits in the market with valuation of $10 per pound, while there is no rabbit seller in the market (no rabbit selling order) or all of rabbit sellers want to sell rabbit with price higher than $10 per pound, this customer's buying order will fail in being finished.


(a) State 1 (b) State 2 (c) State 3 (d) State 4 (e) State 5

**Fig. 1.** This figure shows an example to illustrate how classic market works in a time slot with three orders coming in. Five subfigures respectively correspond to five different states in this example.

And Figure 1 helps to illustrate details about classic market, which is combined with 5 states. First of all, Figure 1(a) shows in state 1 the classic market is initialized with six transaction orders, which might be left from the previous time slot. The order is organized with a specific format, e.g. "Amy [15]: $105" means a seller named as Amy, wants to sell 15 units of goods with valuation $105, where valuation denotes the minimum money Amy are eager to sell her goods; while "Kenny [5]: $90" means a buyer named Kenny wants to buy 5 units of goods with valuation $90, i.e., only when some sellers leave orders which say they can sell goods with the price lower or equal than $90, could Kenny buy goods from them. And there should be a gap between sellers' orders and buyers' orders, just as the red dashed line presents, otherwise these orders will be done in the previous time slot. Figure 1(b) shows how classic market process this new order, when Edward comes into market and launches a buy order with valuation $105. According to the classic market rules, market will let buyers to buy goods with minimum price. Therefore, Edward will firstly buy 5 units of goods from Stan, then 20 units of goods from Bob and finally 7 units from Amy. And his expenditure should be $5 \times 100 + 20 \times 102 + 7 \times 105$, i.e. totally $3275. In Figure 1(c), i.e., state 3, another customer called Ben, who also is the final customer in this time slot we assume, comes into the market and tries to sell 18 units

of goods. Similarly, he will sell all his goods to Cartman as Cartman offers the highest purchasing price in the market. And Ben can thus gain $18 \times 98$, i.e. \$1764. In Figure 1(d), i.e., state 4, Joe comes in market with a sell order of valuation \$100. However, in this moment, there is no buyer who wants to accept this offer. Hence, transaction order proposed by Joe cannot be satisfied. State 5 is the final state of classic market in this time slot. As shown in Figure 1(e), there are still five orders remaining. As Amy and Cartman's order is partially been satisfied, we call these two orders *partially failed* at current time slot. While as poor Joe, Kyle and Kenny's orders remain in deep freeze, we call their orders *totally failed* in this time slot.

## 2 The Framework of Simulator

In this section, we will illustrate the construction of the framework of our simulator, which mainly consists of four classes – *Smart Token*, *Customer*, *Bancor Market* and *Classic Market*. And we will introduce the function of these four classes respectively.

### 2.1 Smart Token

**Algorithm 1** Trading algorithm in Smart Token Class

**Input:** $n$: the tokens' number customer wants to buy or sell

**Output:** $n'$: the tokens' number customer receive – when buying, returns smart tokens' number while when selling, returns reserve tokens'.

1: **if** Customers buy smart tokens **then**
2: Let $n' := \text{round}(S \times \left[\left(\frac{B+n}{B}\right)^{\text{CRR}} - 1\right])$,
where $S$ denotes supply, $B$ means reserve balance and CRR corresponds to the constant reserve ratio.
3: Update the supply for smart token class, let $S := S + n'$.
4: Update the reserve balance for smart token class, let $B := B + n$.
5: Update the smart token's price $p$, let $p := \frac{B}{S \times \text{CRR}}$.
6: **return** $n'$
7: **else**
8: Let $n' := \text{round}(B \times \left[1 - \left(\frac{B-n}{B}\right)^{\frac{1}{\text{CRR}}}\right])$,
where $S$ denotes supply, $B$ means reserve balance and CRR corresponds to the constant reserve ratio.
9: Update the supply for Smart Token class, let $S := S - n$.
10: Update the reserve balance for Smart Token class, let $B := B - n'$.
11: Update the smart token's price $p$, let $p := \frac{B}{S \times \text{CRR}}$.
12: **return** $n'$
13: **end if**

The Smart Token class implements the key idea of Bancor protocol, including the calculation of smart token's price and the number of issued or destroyed smart token's number. In fact, the whitepaper of Bancor protocol [1] hides the calculating formula for specifying how many smart tokens or reserve tokens

should be issued in purchasing or selling. And we find these calculating methods from Bancor protocol's source code [3], which we list below:

$$I_s =< S \times \left[ \left( \frac{B + \Delta_p}{B} \right)^{\text{CRR}} - 1 \right] >,$$

$$I_r =< B \times \left[ 1 - \left( \frac{B - \Delta_s}{B} \right)^{\frac{1}{\text{CRR}}} \right] >,$$

where $I_s$ and $I_r$ denote the number of issued smart tokens and reserve tokens, $B$ means the reserve balance, CRR represents the constant reserve ratio, $\Delta_p$ and $\Delta_d$ correspond to the reserve tokens' number buyer uses to purchase smart tokens and the smart token number seller uses to sell respectively, and symbol $<>$ means the round function, which indicates that the number of issued tokens must be integer according to the Bancor protocol.

Here in Algorithm 1, we show algorithm of buying and selling smart tokens in Smart Token class.

### 2.2 Customer

The Customer class simulates customer's behavior, which offers customer ability of knowing the smart token's price in market, launching and canceling their orders to the market, i.e., Bancor Market class or Classic Market class. Here, we assume that in every time slot, customers in the market will change their valuations of smart token. And if their valuations are larger than smart token's current price, they would like to buy smart tokens, yet be eager to sell if valuations are smaller.

Here, we present the algorithm for customer to change his valuation in Algorithm 2.

**Algorithm 2** Changing valuation algorithm in Customer class

**Input:** $v'$: new valuation

1: Change the valuation $v$ of smart token in Customer class, let $v := v'$.
2: Tell market to cancel the old transaction order: cancelOrder().
3: Get the current smart token's price $p_c$ from asking market, let $p_c$ :=market.getCurrentPrice().
4: **if** $v > p_c$ and customer's reserve balance $b_r > 0$ **then**
5: Launch a new buy transaction order to market, call market.buy().
6: **else**
7: **if** $v < p_c$ and customer's smart token balance $b_t > 0$ **then**
8: Launch a new sell transaction order to market, call market.sell().
9: **else**
10: Do nothing and return.
11: **end if**
12: **end if**

### 2.3 Bancor Market

The Bancor Market class simulates the market which handles customers' requests according to Bancor protocol, just as we introduced in subsection 1.2. In the beginning of every time slot, Bancor market will update the current market price which will be broadcasted within customers about the price of smart token, and helps customers to make valuations.

Here in Algorithm 3 and 4, we show how bancor market deals with the customers' requests, including cancellation and transaction. And in simulating, to simplify the model, we assume that at one time, one customer can only push one order to market.

**Algorithm 3** Process the Cancellation in Bancor Market

**Input:** $\mathbb{C}$: the customer who wants to cancel the transaction order.

**for** Every order $O$ in order list maintained in Bancor Market class: **do**
  **if** $\mathbb{C}$ is $O$'s owner **then**
    Pop $O$ from the order list.
    Break the loop.
  **end if**
**end for**

**Algorithm 4** Pocess the Transaction Orders in Bancor Market

**Input:** $\mathbb{C}$: the customer who wants to cancel the transaction order, $a_t$: the amount of transaction value.

Verify whether the $a_t$ is legal, otherwise throw error – check whether it is integer, and larger than 0, and etc.
**if** $\mathbb{C}$ is launching a buy order **then**
  **if** $\mathbb{C}$'s valuation larger or equal to smart token's current price **then**
    Call smart token class to buy $a_t$ smart token by Algorithm 1, get recerived smart tokens' number as $a_r$.
    Add $\mathbb{C}$'s smart tokens' number, call $\mathbb{C}$.changeTokenBalance($a_r$).
    Decrease $\mathbb{C}$'s reserve tokens' number, call $\mathbb{C}$.changeReserveBalance(-$a_t$).
    Update the order list maintained by market class, since the price of smart token now is changed, and some previous transactions could now be finished.
  **end if**
**else**
  **if** $\mathbb{C}$'s valuation smaller or equal to smart token's current price **then**
    Call smart token class to sell $a_t$ smart token by Algorithm 1, get recerived reserve tokens' number as $a'_r$.
    Add $\mathbb{C}$'s reserve tokens' number, call $\mathbb{C}$.changeReserveBalance($a'_r$).
    Decrease $\mathbb{C}$'s smart tokens' number, call $\mathbb{C}$.changeTokenBalance(-$a_t$).
    Update the order list maintained by market class, since the price of smart token now is changed, and some previous transactions could now be finished.
  **end if**
**end if**

### 2.4 Classic Market

**Algorithm 5** Pocess the Transaction Orders in Classic Market

**Input:** $\mathbb{C}$: the customer who wants to cancel the transaction order, $a_t$: the amount of transaction value.

1: Verify whether the $a_t$ is legal, otherwise throw error – check whether it is integer, and larger than 0, and etc.
2: **if** $\mathbb{C}$ is launching a buy order **then**
3: Make a new buy order which can be saved in order list maintained by market class, by combining parameters such as $\mathbb{C}$ and $a_t$.
4: Update the order list, call updateOrderList() function, and transfer the new buy order to this function.
5: **else**
6: Make a new sell order which can be saved in order list maintained by market class, by combining parameters such as $\mathbb{C}$ and $a_t$.
7: Update the order list, call updateOrderList() function, and transfer the new sell order to this function.
8: **end if**

The classic market class simulates the market running under classic market laws which we have presented in subsection 1.3. Here, we show how classic market processes customers' transaction orders in Algorithm 5 and 6, which, compared with the Bancor market, is more complicated since when market managing transaction orders, it needs to search for whether there are matched orders in order list maintained in classic market.

## 3 Modeling and Operating of Simulator

In this section, we present the model of market simulating and how to use our simulator. Specifically, in subsection 3.1, we will propose the model we construct to simulate both the Bancor market and classic market. And in subsection 3.2 we will show how to use our simulator in codes and simulating algorithms.

### 3.1 Market Simulating Model

In real world, there might be hundreds of thousands of customers swarming into market simultaneously. Therefore, in our simulator, we should allow multiple customers come into market and launch orders in the same time. And due to the need of predicting the future condition of market, the market should be able to evolve while simulating, which means the market is supposed to response the customers' requests and vary in different time period. In this case, we purpose an evolving market model with multiple time slots. And in every time slot, at the beginning, market will send customers the price of goods, i.e., smart tokens. And then, according to the information given by market, customers can launch transaction orders and push these orders to market simultaneously. Then, market will process these orders and update the state of customers' classes as well as the smart token class correspondingly. Finally, the market will maintain its order

**Algorithm 6** Update the Order List in Classic Market

**Input:** $\mathbb{O}$: new order which contained parameters such as customer $\mathbb{C}$, transaction amount $a_t$ and flag $f$ which decide whether this order is to buy or to sell.

**Temporal Parameters:** $L_s$: a temporary list which contains all matched sell orders, $L_b$: a temporary list which contains all matched buy orders.

**if** the order list maintained in classic market class $L$ is empty **then**
  Append $\mathbb{O}$ to $L$ and return, since there is no matched order in $L$.
**end if**
**if** $\mathbb{O}.f$ is BUY **then**
  **for** all other orders $\mathbb{O}_l$ in $L$ **do**
    **if** $\mathbb{O}_l.f$ is SELL and $\mathbb{O}_l.\mathbb{C}$ has lower valuation than $\mathbb{O}.\mathbb{C}$'s **then**
      Add $\mathbb{O}_l$ into $L_s$, which is initialized to be empty.
    **end if**
  **end for**
  **if** $L_s$ is empty **then**
    Append $\mathbb{O}$ to $L$ and return, since there is no matched order.
  **else**
    Sort $L_s$ by sellers' valuation from low to high, as goods with low price are preferred.
    **for** all orders $\mathbb{O}_{l_s}$ in sorted $L_s$ **do**
      **if** $\mathbb{O}.a_t$ is smaller than maximum amount $a_{\max}$ $\mathbb{O}_{l_s}$ could offer **then**
        Update $\mathbb{O}.\mathbb{C}$'s information, by decreasing $a_t$ reserve tokens and decreasing corresponding amount of smart tokens, i.e., $<\frac{a_t}{v_s}>$, where $v_s$ is $\mathbb{O}_{l_s}.\mathbb{C}$'s valuation.
        Update $\mathbb{O}_{l_s}$'s information, decrease $<\frac{a_t}{v_s}>$ smart tokens it provides to sell.
        Update $\mathbb{O}_{l_s}.\mathbb{C}$'s information, by adding $a_t$ reserve tokens and decreasing $<\frac{a_t}{v_s}>$ reserve tokens.
        Break the loop.
      **else**
        Update $\mathbb{O}.\mathbb{C}$'s information, by decreasing $a_{\max}$ reserve tokens and adding $<\frac{a_{\max}}{v_s}>$ smart tokens.
        Update $\mathbb{O}_{l_s}.\mathbb{C}$'s information, by adding $a_{\max}$ reserve tokens and decreasing $<\frac{a_{\max}}{v_s}>$ smart tokens.
        Pop the $\mathbb{O}_{l_s}$ from $L_s$, and remove this order from $L$.
        Update $\mathbb{O}$'s information by decreasing $a_{\max}$ reserve tokens from its transaction value, i.e., $a_t := a_t - a_{\max}$.
      **end if**
    **end for**
    **if** $\mathbb{O}.a_t$ is still larger than 0 **then**
      Append $\mathbb{O}$ to $L$ and return.
    **end if**
  **end if**
**else**
  .....
  {Do same things for new sell order.
  Create $L_b$ but sort is from high to low, since high purchasing price is preferred by seller.}
**end if**

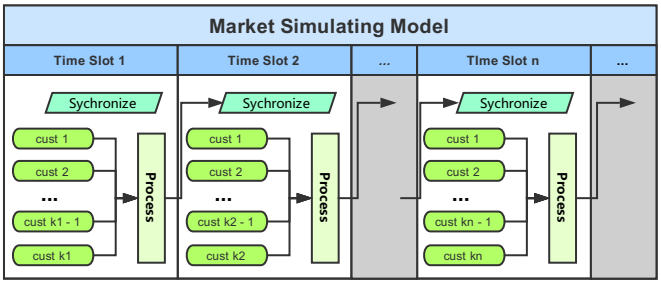



**Fig. 2.** This figure shows how our market model evolves with the time, while in time slot $i$, number of $k_i$ customers come into market and launch orders.

list and wait for the start of next time slot. Figure 2 presents the overview of our market simulating model, where in every time slot, we record the price of smart token, number of launched transactions and failed transactions as indexes for analysis.

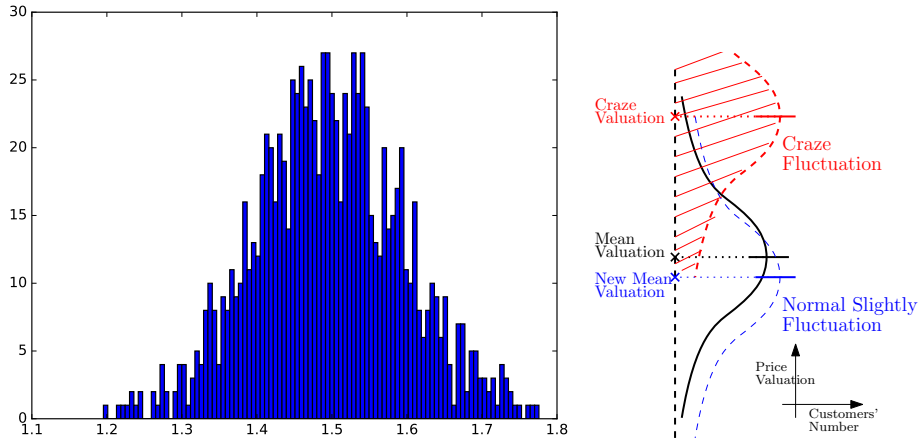
For customers entering the market and launching transaction orders, there are three stipulations in our model.

**(1) One time one order**: In our model, for simplicity, we stipulate that one customer at one certain time slot, can only lanch one order in the market. The reason for this is that we can thus uniformly manipulate every customer's behavior and easily track the launched transaction orders' number for later analysis.

**(2) Gaussian-like distributed number of customers**: We assume that the number of customers approaching market is Gaussian-like distributed with their valuations of smart token's price. Figure 3(a) shows the instance of Gaussian-like customers' number distribution. This assumption is reasonable as in a normal real-world market, most customers' valuation of a commodity is not far from each other, which can be presented in Gaussian distribution. *Here might need more specific argument...*

**(3) All-in and Half-in**: We set two policies to determine the transaction value of orders. The first is that customers use all of their smart tokens or reserve tokens to sell or to buy, namely all-in policy. And the second is that customers leverage half of their assets to sell or to buy, which is called as half-in policy. These two policies also are adopted for simplicity and ensure the reproductivity of our codes. Results show that these two policies actually could lead really different results in classic market, which is shown in section 4.

Further, in our model, based on Gaussian-like distributed customers' number, we develop a method which can simulate the market craze that suddenly almost everyone in market wants to buy or sell the goods, i.e. smart tokens. In normal cases, we assume the base price valuation, i.e. $\mu$ in Gaussian function is exactly the current market price. Therefore, there will be almost 50% buyers and 50% sellers divided by market price, which obeys the normal market discipline.




(a) Gaussian-like Distributed Customers' Number (b) Craze Simulating

**Fig. 3.** Figure 3(a) shows an example of Gaussian-like distributed customers number with their price valuation, where x-axis means the customer's valuation and y-axis represents the customers' number. In this example, there are 1000 customers in the market, indicated by the sum of length of blue bars, who make valuation of smart token based on the base price of 1.5. Figure 3(b) presents an instance of market craze simulation by setting the mean valuation, i.e., $\mu$ in Gaussian function random. The red shadowed space represents customers who launch buy orders after craze valuation goes high.

For instance, in Figure 3(b), the Gaussian curves in black and blue color are equally divided into two parts based on the mean valuation. And according to the Algorithm 2 in customer class, when valuation is larger than mean valuation, i.e., current market price, customers launch buy order; when valuation is smaller than mean valuation, customers choose to sell. Therefore, numbers of buyers and sellers are almost same by the Guassian distribution in normal cases. However, when market craze emerges, we randomly choose the craze valuation in range $(\frac{1}{R}p, Rp)$, where $R$ is a parameter which controls the craze's degree, and $p$ is the current market price. As shown in Figure 3(b), when mean valuation bounces to a relatively high price, the huge skewness of choosing to buy or to sell takes place. And nearly all customers launch buy orders, labelled by red shadow area in Figure 3(b). Thus, the market craze can be well modeled and simulated.

### 3.2 Operating Algorithm

In this part, we talk about detailed algorithms which achieve the modeling in subsection 3.1. Also, we briefly show the parameters in our simulator which are used to initialize the whole model. And by observing experiment results with altering these parameters, we make analysis in section 4.

**(1) Introduction of Parameters:** In modeling, we launch experiments mainly based on four parameters, which is introduced in Table 1, where $N_c$ determines market size and the money amount holds in customers' hand (initialized total amount of smart tokens and reserve tokens), $T$ and $R$ represent the

**Table 1.** Parameters for Simulating Experiments

| Parameters | Definations |
|---|---|
| $N_c$ | Number of customers who make deals in market |
| $T$ | Time interval between market crazes, measured by count of time slots |
| $R$ | Valuation bouncing range parameter of market craze |
| $\sigma_0$ | Variable in Gaussian function for generating customer number's distribution |

frequency of the emergence of market craze and its severe degree respectively, and $\sigma_0$ reflects the steepness of Gaussian curve for distribution of customers' number, which can be viewed in Figure 4.

By controlling these parameters to simulate the performance of market, we actually can view the impact from market's size, market craze and customers' valuation choice.

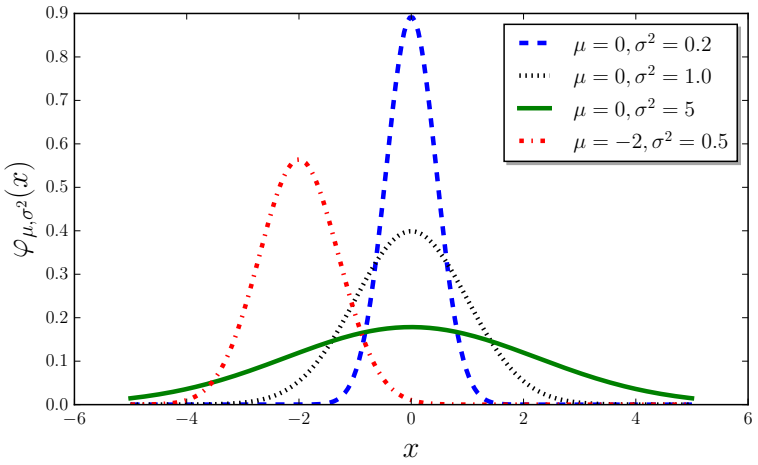



**Fig. 4.** This figure shows normalized Gaussian curves with expected value $\mu$ and variance $\sigma$. In this figure, peak is presented by Gaussian function, where the Gaussian curve climbs to the maximum when $x = \mu$. The curve becomes smoother, i.e., has lower steepness with the growth of $\sigma$; while in contrast, the peak of Gaussian curve is steeper when $\sigma$ being smaller.

**(2) The Algorithm for Bancor Market Simulating:** The algorithm of Bancor market's simulation is presented in Algorithm 7. In real codes, we use the pseudo-random seeds from 1 to 10 to ensure that our experiments can be reproduced.

**(3) The Algorithm for Classic Market Simulating:** The algorithm of Bancor market's simulation is presented in Algorithm 8, which is almost similar with the Algorithm 7 with the experimental principle of controlling differences. Also, what should be noticed is that in classic market, there is no smart tokens be issued or destroyed in transactions. And customers always generate valuations based on a fixed constant.

**Algorithm 7** Algorithm for Bancor Maket's Simulation

**Input:** $N_t$, $N_c$, $T$, $R$ and $\sigma_0$ which have been introduced in Table 1.

Initialize the input parameters.
Initialize a smart token, such as $\mathbb{S}$=Smartcoin(*name*='KennyCoin', *reservetokenName*='ETH', *CRR*=0.2, *Price*=1, *IssueNum*=initIssue).
Initialize a Bancor Market which uses smart token we have just initialized, such as $\mathbb{M}$=BancorMarket(*smartToken*=$\mathbb{S}$)
Initialize customers with $N_c$ number, such as Customer(*smartToken*=$\mathbb{S}$, *market*=$\mathbb{M}$, *tokenBalance*≈200, *reserveBalance*≈200)
**for** every time slot $t$ in $N_t$ **do**
    Sychronize the $\mathbb{M}$ by $t$. {Ask market to generate the current market price}
    **if** $t$ is not the multiple of $T$ **then**
        Generate valuation array, which makes number of customers Gaussian-like distributed by current market price $p$ as $\mu$, and $\sigma_0$ as $\sigma$ in Gaussian function.
        Use valuation array generated to call changeValuation() for every customer – as shown in Algorithm 2, by which customers actually also launch orders.
    **else**
        Simulate market craze, and generate valuation array by randomly select mean valuation in range $(\frac{1}{R}p, Rp)$ as $\mu$, and $\sigma_0$ as $\sigma$ in Gaussian function.
        Use valuation array generated to call changeValuation() for every customer – as shown in Algorithm 2.
    **end if**
    Collect simulating data for future analysis.
**end for**

**Algorithm 8** Algorithm for Bancor Maket's Simulation

**Input:** $N_t$, $N_c$, $T$, $R$ and $\sigma_0$ which have been introduced in Table 1.

Initialize the input parameters.
Initialize a smart token, such as $\mathbb{S}$=Smartcoin(*name*='KennyCoin', *reservetokenName*='ETH', *CRR*=0.2, *Price*=1, *IssueNum*=initIssue).
Initialize a Bancor Market which uses smart token we have just initialized, such as $\mathbb{M}$=ClassicMarket(*smartToken*=$\mathbb{S}$)
Initialize customers with $N_c$ number, such as Customer(*smartToken*=$\mathbb{S}$, *market*=$\mathbb{M}$, *tokenBalance*≈200, *reserveBalance*≈200)
**for** every time slot $t$ in $N_t$ **do**
    Sychronize the $\mathbb{M}$ by $t$. {Ask market to generate the current market price}
    **if** $t$ is not the multiple of $T$ **then**
        Generate valuation array, which makes number of customers Gaussian-like distributed by current market price $p$ as $\mu$, and $\sigma_0$ as $\sigma$ in Gaussian function.
        Use valuation array generated to call changeValuation() for every customer – as shown in Algorithm 2, by which customers actually also launch orders.
    **else**
        Simulate market craze, and generate valuation array by randomly select mean valuation in range $(\frac{1}{R}p, Rp)$ as $\mu$, and $\sigma_0$ as $\sigma$ in Gaussian function.
        Use valuation array generated to call changeValuation() for every customer – as shown in Algorithm 2.
    **end if**
    Collect simulating data for future analysis.
**end for**

# 4 The Analysis of Results

In this section we present experiment results of simulation, which includes comparing the market performance under Bancor protocol and classic marketing rules. First of all, we will introduce indexes which are leveraged to reflect and measure the market performance. And then, based on these indexes and parameters introduced in Table 1, we launch detailed analysis.

## 4.1 Indexes for Measuring Market Performance

Here we introduce several indexes which are used to illustrate whether market performs well under specific laws. And these indexes can be mainly divided into two categories, i.e., *price-oriented* and *transaction-oriented*.

**(1) Price-oriented Indexes:** Under most circumstances, a healthy market is supposed to possess currency with considerably stable price. Therefore, it is necessary to track the price of smart token in Bancor market to see how it fluctuates and responds for the market craze. Besides, we use *price slipping ratio* to evaluate the fluctuating degree of smart token's price, which records the ratio of number of time slots in which price drops at a certain rate to number of all time slots. In fact, faced with high possibility of price slipping of smart token in market, i.e., high price slipping ratio, customers are easy to be in a panic which might spawn terrible problems.

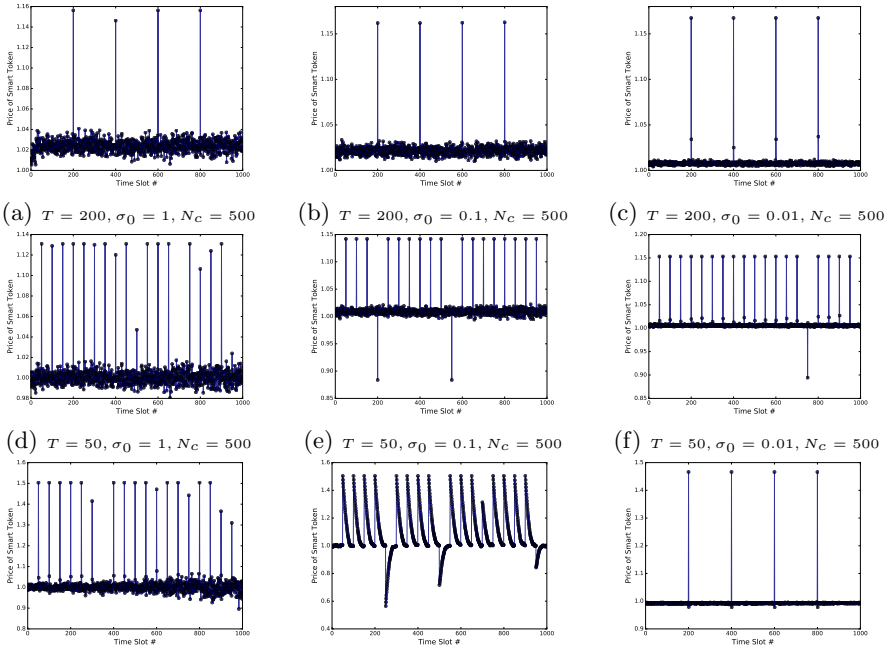
**(2) Transaction-oriented Indexes:** One of the main superiorities claimed by Bancor protocol is that it solves the problem of "Double Coincidence of Wants" that compared to classic market, Bancor market allows transaction to be finished whether there are matched orders or not, and thereby largely increasing liquidity of market. However, this advantage is not be verified by experiments and as we argued in subsection 1.1, transactions in Bancor market may also be foundered to be processed. Therefore, here we record the *total transactions' number* and *transactions' cancel ratio* (simulator requires in every time slot customers has to cancel their transaction orders if they are not finished) of both Bancor market and classic market to see whether Bancor protocol efficiently handle the problem of "Double Coincidence of Wants" and largely improve the market's liquidity.

## 4.2 Analysis Based on Price

The price of smart token fluctuates after every transaction under bancor protocol. Since it happens too frequently, we only track the price of smart token in the end of every time slot. Figure 5 represents several price fluctuating examples under different parameter initializations. By which, we can view that though market craze merges which makes the price of smart token changes fiercely, the Bancor market actually is able to adjust the price to a relatively stable state. And by comparing subfigures in Figure 5, with the decrease of $\sigma$ and the increase of $N_c$, the price of smart token could be more stable.

However, since Figure 5 only draws the results based on one experiment with the pseudo-random seed 0, which, to some extent, is unrepresentative to reflect

the uniform results. Besides, only observing the price fluctuation of smart token by figures is not accurate enough. Therefore, we quantify the price fluctuating degree by slipping ratio and analyze it by repeating experiments for 10 times with same parameter pairs and calculating the average data. We divide price slipping into two kinds – *medium slip* and *huge slip*, where the former one denotes the cases price decreasing rate exceeds 5%, and the later one denotes cases exceeds 20%. In Figure 6, we visualize the price slipping ratio results by bar charts. In Figure 6(a), we can find that with the growth of valuation bouncing range $R$, the price splitting ratio slightly increases. And Figure 6(b) shows when market craze interval $T$ decreasing, the splitting ratio can be largely improved. And in Figure 6(c), it can be viewed that the splitting ratio drops when customers' number $N_c$ decreases, i.e., market has smaller size. Also, among Figure 6(a), (b) and (c), with the increase of $\sigma$, the price splitting ratio, in different degree, is improved.



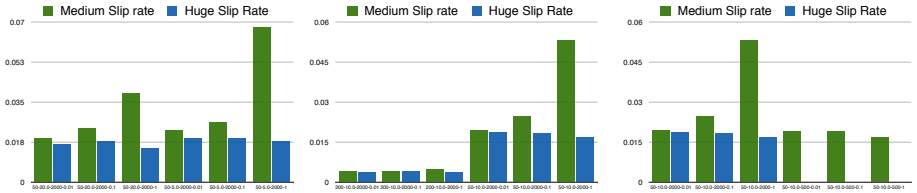

(a) $T = 200, \sigma_0 = 1, N_c = 500$ (b) $T = 200, \sigma_0 = 0.1, N_c = 500$ (c) $T = 200, \sigma_0 = 0.01, N_c = 500$

(d) $T = 50, \sigma_0 = 1, N_c = 500$ (e) $T = 50, \sigma_0 = 0.1, N_c = 500$ (f) $T = 50, \sigma_0 = 0.01, N_c = 500$

(g) $T = 50, \sigma_0 = 1, N_c = 2000$ (h) $T = 50, \sigma_0 = 0.01, N_c = 2000$ (i) $T = 200, \sigma_0 = 0.01, N_c = 2000$

**Fig. 5.** This figure shows the price fluctuation in Bancor market simualtion initialized with $N_t = 1000$, $R = 10$ and corresponding $N_c, T, \sigma_0$ in subfigures' caption, with the pseudo-random seed 0 and all-in policy. In this figure, every blue mark represents the price of smart token in the end of the current time slot, and blue lines track the trend of price's fluctuation.

In fact, as results demonstrate, when market crazes emerges considerably often, and the market owns a large size, the price slipping ratio can be very high, e.g. almost 7% in Figure 6(a), which means **the price actually can fluctuate significantly in Bancor market**. This might lead many negative effects.




(a) $R = 20$ vs. $R = 5$ (b) $T = 200$ vs. $T = 50$ (c) $N_c = 2000$ vs. $N_c = 500$

**Fig. 6.** This figure shows the price slipping ratio in 1000 time slots with different parameters settings. The labels on x-axis in the figure means the specific parameters pair, which is organized as $T$-$R$-$N_c$-$\sigma_0$; the value on y-axis reflects the splitting ratio.

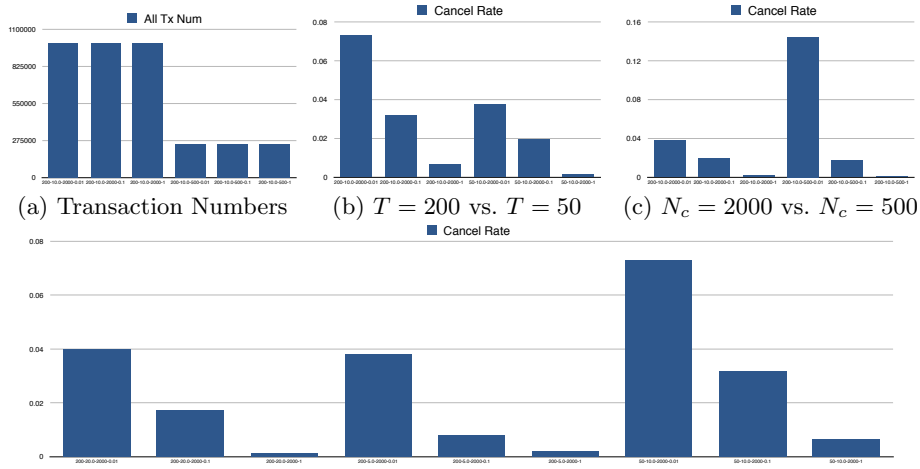
### 4.3 Analysis Based on Transaction

In real-world market, when customer proposes a new order, he must pay a certain amount fee to the market. And customers' benefits can be harmed if their orders cannot finally be finished in market while wasting money on useless transaction fee. Therefore, the efficiency and accuracy of market dealing with multiple transaction orders is more than essential. Here, we calculate the launched transactions' number, and transactions' cancel rate as well as failed rate as matrices to evaluate the market's performance. And we use classic market as control group to view whether Bancor protocol does improve market's efficiency and liquidity.

**(1) Observation on Bancor Market:** First of all, we observe the Bancor market's performance on dealing with transactions. Figure 7 shows transaction number and cancel ratio under all-in policy in Bancor market. Figure 7(a) presents that by bancor protocol, with specific customers' number $N_c$, the launched transactions hold with a stable number, which indicates most of customers in market smoothly launch orders and always hold some smart tokens or reserve tokens under all-in policy. In Figure 7(b), with the increase of $T$, the transations' cancel rate improves, which means more useless transaction fee will be payed by customers. Figure 7(c) illustrates when the market owns fewer customers, i.e., fewer money in customers hands and fewer transactions numbers as shown in Figure 7(c), the transaction's cancel rate will improve significantly to even more than 15%. And by Figure7(d), we know that the cancel rate goes high when craze range $R$ rises to a certain constant, but does not keep increasing or decreasing with the rising of $R$. Also, by Figure 7(b), (c) and (d), with smaller $\sigma_0$, the cancel rate sticks to higher place.

By these plots in Figure 7, we learn that with the small $N_c$, i.e., small market size which actually represents the current status of most virtual current markets, and small $\sigma_0$, i.e., the closer valuation between customers, **the failure transaction orders can even take over more than 10% in Bancor market**, which is actually intolerable in real world.

**(2) Observation on Classic Market:** Then, we study the classic market's performance under all-in policy. In classic market, the cancel rate of transaction in fact can be divided into two kinds as one case the transaction might be partially satisfied before being canceled (Cartman and Alice's case in Figure 1) and another the transaction might be totally ignored by market and then be

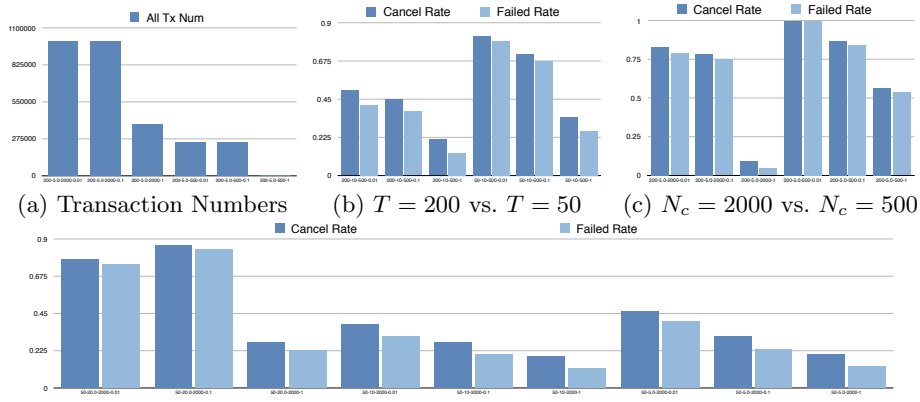



(a) Transaction Numbers (b) $T = 200$ vs. $T = 50$ (c) $N_c = 2000$ vs. $N_c = 500$

(d) $R = 20$ vs. $R = 5$ vs. $R = 10$

**Fig. 7.** This figure shows the transaction-oriented indexes in Bancor market in 1000 time slots with different parameters settings. The labels on x-axis in the figure means the specific parameters pair, which is organized as $T$-$R$-$N_c$-$\sigma_0$.

canceled (Joe and Kenny's case in Figure 1). And we call the former one *cancel rate*, and the later one *failed rate*. Figure 8 shows the results.




(a) Transaction Numbers (b) $T = 200$ vs. $T = 50$ (c) $N_c = 2000$ vs. $N_c = 500$

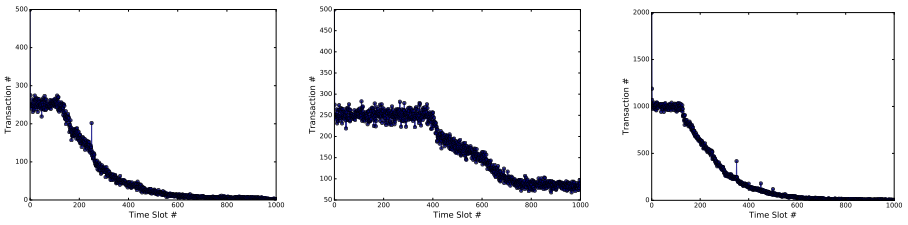
(d) $R = 20$ vs. $R = 10$ vs. $R = 5$

**Fig. 8.** This figure shows the transaction-oriented indexes in Classic market in 1000 time slots with different parameters settings under all-in policy. The labels on x-axis in the figure means the specific parameters pair, which is organized as $T$-$R$-$N_c$-$\sigma_0$.

In Figure 8(a), the launched transactions' number in classic market varies between different parameter settings while the Bancor market's transactions' number stays stable with specific $N_c$. This is because under all-in policy in classic market, some customers quickly run out their assets of smart token and reserve tokens as they generate low valuation to sell and generate high valuation to buy with all they own. And then transacion numbers will decrease as smaller amount of customers in market owns money for transaction. And Figure 9 shows how the transactions number decreases under all-in policy – with more frequent

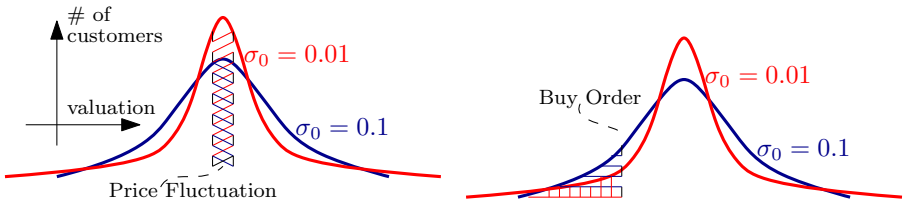
market craze, i.e., smaller $T$, and larger market size, i.e., larger $N_c$, transactions' number is apt to decreases more rapidly.

Figure 8(b) shows that with more market craze, i.e., smaller $T$, both transactions' cancel and failed rates are increasing. And Figure 8(c) presents with larger market size – larger $N_c$, transactions are more likely to be finished in low failure rate. In Figure 8(d), the transactions' cancel and failed rates go to low when $R$ is 10, just like Bancor market. Also, similar with Bancor market, when $\sigma_0$ is smaller, both being canceled and failed transactions' ratios rise significantly – this phenomenon actually stems from our assumption of Gaussian-like distributed customers' number, which we explain by Figure 10.




(a) $T = 50, N_c = 500$ (b) $T = 200, N_c = 500$ (d) $T = 50, N_c = 2000$

**Fig. 9.** This figure shows the launched transactions' number in Classic market in 1000 time slots with $R = 10$ and $\sigma_0 = 0.01$.

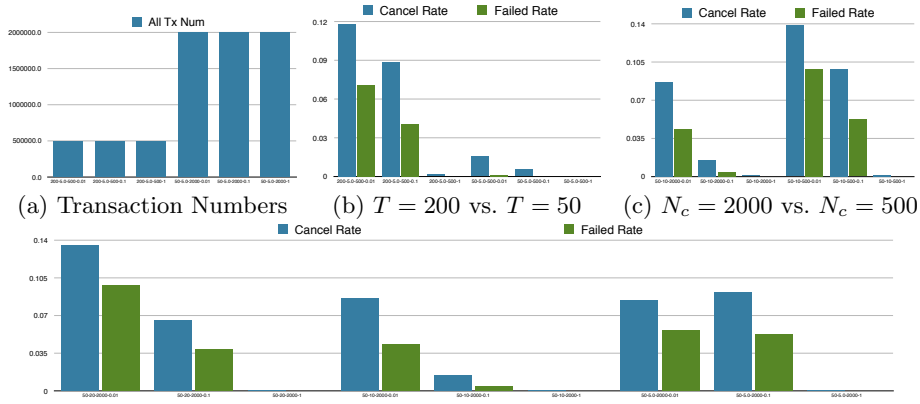



(a) Bancor Market (b) Classic Market

**Fig. 10.** This figure helps to explain why $\sigma_0$ impacts transactions' cancel or failed rate in Bancor and classic market.

In Bancor market, just as Figure 10(a) shown, when price of smart token fluctuates slightly, when $\sigma_0$ is small, e.g., $\sigma_0 = 0.01$, the number of influenced customers is larger than case when $\sigma_0$ is large, e.g., $\sigma_0 = 0.1$ – red dash shadowed area is larger than blue shadowed area in Figure 10(a). Hence, much more transaction might be failed when $\sigma_0$ is smaller in Bancor market. Similarly in classic market, presented by Figure 10(b), when a buy order, which purchase smart tokens with lower price than its valuation comes, the the number of customers who are qualified to satisfy this buy order is smaller when $\sigma_0 = 0.01$ than case when when $\sigma_0 = 0.1$ – red dash shadowed area is smaller than blue shadowed area in Figure 10(b). Thus, the cancel or failed rate will be raised when $\sigma_0$ is small.

Further, by high transaction cancel and failed rates in Figure 8 in which many parameter settings lead failed rate over 50%, we can conclude that **under all-in policy, the "Double Coincidence of Wants" problem does exist and could bring disastrous performance of market in efficiency.**




(a) Transaction Numbers (b) $T = 200$ vs. $T = 50$ (c) $N_c = 2000$ vs. $N_c = 500$

(d) $R = 20$ vs. $R = 10$ vs. $R = 5$

**Fig. 11.** This figure shows the transaction-oriented indexes in Classic market in 1000 time slots with different parameters settings under half-in policy. The labels on x-axis in the figure means the specific parameters pair, which is organized as $T$-$R$-$N_c$-$\sigma_0$.

However, in real-market, customers might seldom purchase and sell with all they have. Therefore, we also consider the situation that in every time slot, customers only use half of their assets to buy or sell. Figure 11 shows the transaction-oriented performance in classic market under half-in policy, which is quite different with the performance under Bancor market.

In Figure 11(a), the launched transactions' number becomes stable as customers always hold some smart tokens or reserve tokens. Figure 11(b) shows with the higher frequency of market craze, the cancel and failed rates are decreasing. In Figure 11(c), it can be viewed that with larger market size, transaction are more likely to be smoothly handled. And Figure 11(d) helps to illustrate that with $R = 10$, transactions are more likely to be smoothly processed.

In fact, by our observation, properties we explored under all-in policy still be found under half-in policy. However, in half-in policy, the cancel rate or failed rate is much smaller than those in all-in policy. And by Figure 11 we can see that the largest failed ratio does not exceed 10%, which indicates **the "Double Coincidence of Wants" might no longer be a problem in classic market under half-in policy.**

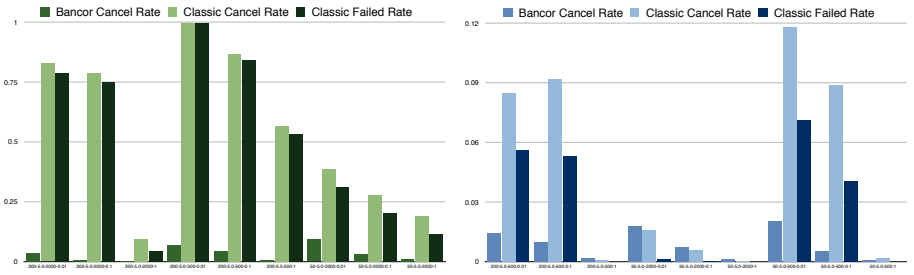
**(3) Bancor Market vs. Classic Market:** By comparing the transaction-oriented performance between Bancor market and classic market, we conclude several properties as below:

**a.** The market craze actually positively accelerates the processing of transaction orders both for Bancor and classic market.

**b.** The increase of market size helps Bancor market and classic market dealing with transaction orders much more smoothly.

**c.** Under all-in policy, Bancor protocol does solve the the problem of "Double Coincidence of Wants" as its performance far exceeds classic market, as shown in Figure 12(a) that Bancor market deals with transaction orders with fewer cancellation; while under half-in policy, the existence of "Double Coinci-

dence of Wants" is uncertain and sometimes classic market performs better than Bancor market, as shown in Figure 12(b) that generally transaction cancel or fail rate is low, and in some parameter settings, classic market owns lower cancel rate.




(a) All-in Policy (b) Half-in Policy

**Fig. 12.** This figure compares the transaction cancel and fail rate between Bancor market and classic market with different parameter settings organized as $T$-$R$-$N_c$-$\sigma_0$. Subfigure (a) shows the all-in policy's case and subfigure (b) presents half-in policy's case.

## 5 Conclusion and Future Work

In conclusion, compared with classic market, market under Bancor protocol does own several superiority, as it helps to eliminate the problem of ... However, in some cases, Bancor protocol is flawed ...

## References

1. Eyal Hertzog, Guy Benartzi & Galia Benartzi. Bancor Protocol, Continuous Liquidity and Asynchronous Price Discovery for Tokens through their Smart Contracts; aka "Smart Tokens" May 30, 2017
2. Sirer, E. G., & Daian, P. (2017, June 19). Bancor Is Flawed. Retrieved July 17, 2017, from http://hackingdistributed.com/2017/06/19/bancor-is-flawed/
3. Levi, Y., Pinhas, I., Manos, B., & Swende, M. H. (2017, July 20). Bancorprotocol/contracts. Retrieved July 20, 2017, from https://github.com/bancorprotocol/contracts