Research on Time Performance of Dynamic Nested Transactions in Open Reconfigurable Network

Jianping Qu[1], Ligang Dong [2], Lingjia Gui [3], Weiming Wang[4], Julong Lan[5]

[1,2,3,4]Zhejiang Gongshang University, Hangzhou310018, China
[5]National Digital Switching System Engineering & Technological Research Center, Zhengzhou450002, China
Email:{qujianping, guilingjia}@pop.zjgsu.edu.cn;{donglg,wmwang}@mail.zjgsu,edu.cn;
ljl@mail.ndsc.corn.cn

***Abstract*——Open Reconfigurable Network (abbreviated as ORN) is a new type of distributed network architecture, which is composed by integrated network management nodes (abbreviated as MN) and more open reconfigurable routing or switching nodes (abbreviated as NN), which are managed by MN. During the path deployment among NN, there probably exists failures and even repeated failures due to the resources occupation and conflict causes, so it is inefficient to use the traditional transaction model. To improve deployment efficiency, this paper focuses on time performance of dynamical nested transactions in the ORN. In this paper, firstly, we proposed the dynamic nested transaction algorithm on the basis of the existing nested transaction; secondly, we established the nested transaction mathematical model in the ORN; finally, we proved the superiority of the model compared with the traditional model through mathematical analysis and made results more intuitive by digital experiment.**

***Index Terms*—Open reconfigurable network, traditional transaction model, nested transactions, dynamic, mathematical model.**

I. INTRODUCTION

In the Open Reconfigurable Network (abbreviated as ORN)[1], the open means that the definition of each module, interface with each other and communication protocols in network equipment, are open to the public, unified and even standardized[2]. Reconfigurable ability refers to that the nodes function in the network system can be dynamically created, deployed, uninstalled, monitored, expanded and upgraded[3]. The realization of reconfigurable performance is based on modular network functions, and it provides flexibility for network nodes. The ORN reconfigures various underlying resources in the network device on the same platform by defining the mutual interface between devices and standardized communication protocols in network, which enables network equipment can realize cross-manufacturer interoperability, compatibility and flexible function reorganization[4]. Management nodes (abbreviated as MN) can control and manage network nodes (abbreviated as NN), such as path deployment—the management nodes control the resource reservation of the network nodes, thus there exists distributed transactions[5][13].

Transaction is a data manipulation sequence defined by users, and its correct execution needs to ensure the ACID (Atomicity, Consistency, Isolation, Durability) characteristics[5]. In the traditional transaction model, a transaction is a flat sequence of operation, and there is no internal structure, but with ACID properties. The two-phase commit protocol is used in transaction processing, so when all the participants can not implement successfully, the transaction must be rolled back. In this transaction model, the previous part of the failure need to be reran, which will inevitably lead to a lot of unnecessary overhead and reduce the efficiency of transaction processing[7].

In the ORN, some MNs manage resource allocation and topology on multiple NNs. Thus when a MN reserves resource in the NN, it often ends up with resource confliction because the NN has been occupied by other MNs, which leads to repeated failure and would greatly increase the transaction's execution time. Because of the low efficiency in the traditional transaction model, people began to apply the nested transaction in real environment. The nested transaction[7][8][9] refers to the transaction which explicitly includes another transaction, the included is called the subtransaction, the containing called its parent transaction, transactions which are at the same level are brother transactions. In nested transactions, the parent transaction contains hundreds of subtransactions, and some subtransactions have the same function, the set of which is called a functional alternative set. The nested transaction's deployment is efficient, because it allows the concurrent execution of internal subtransactions and provides a good control handle for failure. That is, the failure of subtransactions is relatively independent to their brothers, so it can be replaced by its brother (referred to as functional alternative) and the failure of a

Corresponding author: Ligang Dong

doi:10.4304/jcp.8.9.2330-2338

subtransaction will not cause the entire transaction's failure.

In the nested transaction, to find alternative subtransactions in advance would make the MN's preparation time greatly increased, leading to large cost of system time[12]. To solve this problem, we proposed the dynamic nested transaction: we choose the best path for deployment and don't find alternative subtransaction in advance, just when the deployment fails, we look for the subtransaction which has the same function instead. In this paper, we mainly studied how to use dynamic nested transactions in the ORN to improve the efficiency of the execution of the transaction and optimize the time performance.

II. Related Work

A. Classification of Nested Transactions

On the basis of literature, we classified nested transactions from three different angles.

1. The relationship between the subtransaction with the father and subtransaction

(1) Close nested transactions

The nested transaction[5]: A tramsaction can be decomposed into a number of subtransactions, only after the father transaction submitted, then its subtransactions can be successfully submitted and the result would be sent into a permanent storage area. If a father transaction fails, all of its subtransactions must be rescinded. The advantages are: (1) it allows the concurrent execution of the internal subtransactions; (2) the brother transactions are independent, and thus the failure of a subtransaction does not cause the entire transaction's failure.

(2) Open nested transactions

When a subtransaction is successfully submitted, the submission status is visible to other transactions. Before father transaction is submitted, the transaction releases the lock and then the other subtransactions can get it, so the other subtransaction's execution time is earlier, thereby enhancing the efficiency of the execution of the transaction. So the open nested transaction is no longer strictly limited by father transaction, and it prevents the case of the failure of father transaction which leads to the revocation of all the subtransactions[3].

2. Nested transactions on the execution of multiple participants

(1) Nested transactions of unicast deployment

Managers deploy the participants one by one. Thus, in the execution of unicast, there is a problem of priority allocated to each subtransaction[8].

(2) Nested transactions of multicast deployment

Managers deploy all participants synchronously in the deployment process, that is, the deployment requests to all participants are at the same time.

3. The target of nested transactions

(1) Considering the time of nested transactions: During deployment, when we select the way of the deployment, we only consider the time spent factor, regardless of the cost of deployment.

(2) Considering the cost of nested transactions: when the deployment process fails, it also has the cost of deployed nodes and those nodes who have been deployed successfully and submitted also need to compensate for the revoke [11][12].

B. Application of Nested Transactions

This section focuses on the previous nested transaction in a specific environment.

1. Real-time nested transactions

Active real-time database transactions can trigger any depth, and the processing of transactions are very complex. In the processing of a transaction, we should clarify the deadline of the transaction and the dependencies with other transactions[13]. In the traditional transaction processing, real-time transactions wait for the schedule execution after pre-analyzing. If the transaction died for some reason, it fails. The transaction may die again for ultra deadline even if re-scheduling it.

A real-time transaction can have multiple functional alternatives in each execution process. If functional alternative died for some reason and the deadline of the transaction yet to come, we can select another function alternative. As long as there is a functional alternative to be executed, it can be submitted. The introduction of the functional alternative has greatly enhanced the ability of real-time transactions to adapt to the system operating environment and improved the probability of a real-time transaction submitted successfully. Before a real-time transaction participates in the system scheduling, it isolates the functional alternative set, which will avoid additional time overhead and improve the efficiency of transaction execution.

2. Mobile nested transactions

The transaction processing developed from the centralized system and distributed system to the transaction processing of mobile devices, known as the mobile transaction. The transaction issued by the mobile host with timing constraints is called Mobile Real-Time Transaction(MRTT)[14]. Transaction's mobility and wireless network's inherent defects in the Mobile distributed computing environment make it insufficient for the traditional transaction model to describe the mobile real-time transaction with complex structure, while the nested transaction is able to better describe the structure and improves the concurrency of the execution of the subtransaction's root transaction, so it will support the execution of distributed real-time transactions better.

The mobile nested transaction: A Mobile Real-Time Transaction(MRTT) may have several functional alternative sets, each functional alternative set may substitute one or more functional alternative subtransactions, thus forming the nested structure of the transaction. As long as a subtransaction in each functional alternative can be submitted, the MRTT will be submitted. Obviously, the functional alternative in mobile nested transactions increases the reliability of transaction's execution.

III. The Dynamic Nested Transaction Algorithm

A. The Need of Raising Dynamic Nested Transaction Algorithm

According to the execution of nested transaction's subtransaction, we divide the current nested transactions into the following two:

(1) Prepared subtransactions

We look for all the subtransaction that have the same function as the alternative set, but only when the execution of a subtransaction fails, we look for a new subtransaction from the prepared functional alternative set instead.

(2) Concurrently executed subtransactions

All the functional alternative subtransactions execute at the same time, but only a subtransaction can be finally submitted[15]. This method increases the system's overhead, but it saves much time of re-execution for the backup subtransactions in the case of the subtransaction's failure rates are relatively high.

In the way of pre-prepared subtransactions, we need to find all the alternative subtransactions in advance, which will increase MN's preparing time and the system's time. While in the way of concurrently executed subtransactions, all the functional alternative subtransactions execute simultaneously. Although it saves the time of re-execution, but it increases the system's overhead[16]. Therefore, in order to save time cost and overhead, we propose dynamic preparing subtransaction: we select the optimal path for deployment, but we don't find a functional alternative subtransaction in advance, and we just look for the re-run subtransaction with the same function when the execution failed.

B. Dynamic Nested Transaction Algorithm

In the ORN, the calculation of routing table is done in MN and MN distributes the routing table to each NN. To classify the various types of packets by different sender and recipient, we can get the following three representatives: (1) the local network unit MN sends information packets to the local NN; (2) the external network unit MN sends information packets to the local network unit NN; (3) external information packets are sent to the other network element MN by NN routing managed by the MN.

MN deploys the path of NN by the way of unicast and multicast. This paper gives the following definition:

Definition 1 Successfully configured within two rounds: the configuration nodes are successfully submitted in the first round or the configuration of nodes can not be submitted in the first round but all can be successfully submitted in the second round.

Definition 2 The nesting depth: it refers to the number of deployment required in successful deployment of the transaction execution path.

In the ORN, there exists two typical kinds of process in path deployment: routing deployment and resource reservation[17]. Routing deployment is that MN in the network element calculates the routing table and sends routing information to NN, thus completing the deployment of the path. The deployment completes in the form of broadcast and it has high success rate. While resource reservation means that MN uses the resource reservation protocol to send the resource reservation request to each NN, then establishes and maintains the state to provide the requested service in router. The difference are: In routing deployment, when there exists the failed NN and a functional alternative, the routing table of NN's neighbor also changes, so the new deployment path includes two neighbors of the failed NN; while in the resource reservation, the first deployment of two neighbor nodes are still valid, thus eliminating the need for new deployment.

Because the execution of resource reservation easily leads to the lack of resources or conflicts and MN uses the form of unicast, the traditional deployment is less efficient. According to the characteristics of the deployment in the ORN, we come up with the dynamic nested transactions, and the algorithm is described as follows:

Step1: MN determines the feasible path between the two edge NNs and then sends the message for configuration command to NN;

Step2: If NN confirms that it can successfully execute the request, it sends confirmation message to MN;

Step3: If MN receives all "SUCCESS" message of NN, then it sends "COMMIT" message to all NNs; else if MN receives "FAILURE" Message, it rolls back the failed forwarding items, and then retains data configuration of other successfully deployed forwarding items;

Step4: MN will find the two forwarding items which are on the same path with the failed one and next to it, and then look for a feasible path between the two forwarding items: If there exists, then "Step5"; else "Step6;

Step5: MN re-sends the configuration message to the forwarding items on the new path: If MN receives "SUCCESS" message of all NNs, then it sends the "COMMIT" message; else if it receives any "FAILURE" message, then it goes back to "Step4";

Step6: If two forwarding items are edge forwarding items, let MN send "ABORT" message to inform all NNs to revert to the state before the transaction; else if there is at least an edge forwarding item, the forwarding will be treated as a new failure, repeat"Step4".

IV. The Mathematical Model of The Nested Transactions

A. The Dynamic Nested Transaction Model

As described in[13], there exists the following four kinds of dependency between the subtransactions in nested transactions:

(1) Priority relations: If A is prior to B, B must be executed after the execution of A;

(2) Select relations: If A and B are the relationship of two elect one, you do A or B. Two sub-even in the first stage of the transaction have been ready, but in the end there is only one subtransaction can be submitted (Commit), another must be rolled back;

(3) Preference relations: If A and B are the preference

relation between A and B, there exists alternative relationship between A and B. Set the condition such as A is prior to B in the same environment, B is executed after A fails. Preference relation is a special kind of relationship, in practical applications, in order to show the order of execution of subtransactions, we will select relationships into preference relations according to the algorithm;

(4) Equal relations: If A and B are equal relationship, said A and B must eventually be submitted, but their order of execution is arbitrary.

In the ORN, every path deployment is the equivalent of the operation for a transaction, and each node just as a forwarding item NN_i corresponds to a transaction's subtransaction. A transaction named T with the entry NN_{in} and export NN_{out} can be expressed as: $T=\{S_i \mid NN_i \in < NN_{in}, NN_{out} >\}$, S_i is called the set of transaction nodes, NN_i is the subtransaction node, $< NN_{in}, NN_{out} >$ is called the transaction execution path; all the $NN_i \in < NN_{in}, NN_{out} >$ constitute the transaction node collection of T ,just as Figure 1 shows.

The transaction can be expressed as

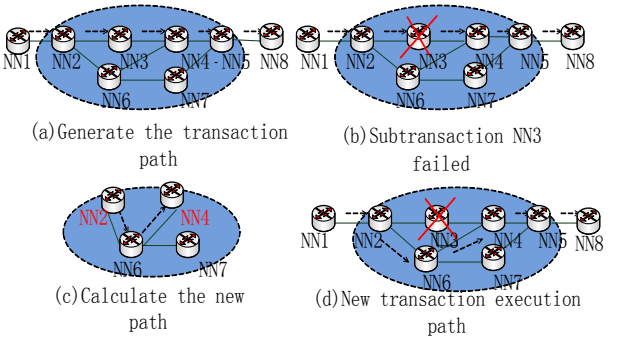

Figure 1. The dynamic nested transaction model

$T=\{S_i \mid NN_i \in < NN_1, NN_8 >\}$, all the subtransactions in the transaction nodes set $\{NN_1, NN_2, NN_3, NN_4, NN_5 NN_8\}$ are equal relations, subtransactions NN_3 and NN_6 are preferences relations, and NN_6 is an alternate subtransaction.

B. *The Analysis of Dynamic Nested Transaction Model*

In order to better adapt to the characteristics of mobile real-time, we come up with the concept of transaction's functional alternative set. As long as a functional substitute is successfully submitted[14], the transaction can be submitted. In the ORN, we use the nested transactions and the function of the functional alternative set, their advantages are as follows:

(1)The success rate of deployment is improved. When the node configuration fails, partial nodes roll back and it looks for an alternative node to reconfigure, which finally reduced the number of nodes that need to reconfigure and improve the efficiency of configuration.

(2)MN's running time significantly is reduced. The deployment path of the traditional transaction is the same as the nested transactions, the nested transactions only deploys those who have not been deployed, and thus in each new deployment, the number of nodes that nested transactions need to deploy is less than the traditional, thereby reducing the running time of MN.

(3)The operating burden of MN is reduced. A MN manages multiple NNs, thus reducing the number of NNs that need to re-deploy and it will reduce the burden of MN's running.

In the ORN, to find all the alternative subtransactions[9] for each NN in advance will greatly increase system deployment's overhead, and it is mainly that the preparing time for management nodes will increase, thus using the way of dynamic preparing subtransaction is optimal. This is something which will be scientifically proven later.

Example 1: The functional alternative set exists

As it is shown in Figure 1, when deployment of NN_3 fails, according to the above algorithm, we will get access to the upstream node NN_2 and the downstream node NN_4, cut off $NN_1 NN_3 NN_5 NN_8$ and the links connected with them, re-calculate $< NN_2, NN_4 >$, and then use NN_6 to replace NN_3. Apparently we can get the new transaction execution path.

Example 2: No functional alternative set

As it is shown in Figure 2, when NN_4 fails, according to the above algorithm, we will get the new transaction execution path as the following Figure 2.

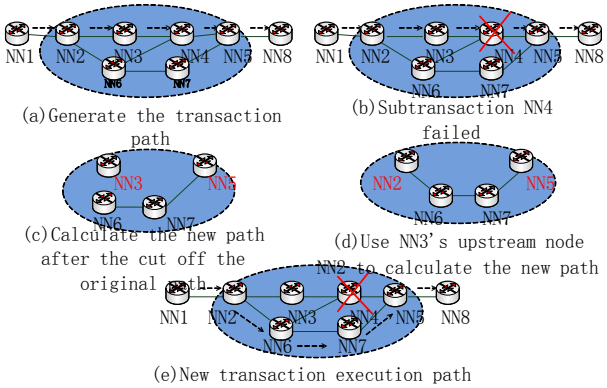

Figure 2. NN4 deployment subtransaction failed

V. Analysis of Nested Transactions' Deployment Time

To compare the performance between traditional deployment and dynamic nested deployment, we analyze two cases of broadcast and multicast. In order to facilitate the expression, Table 1 defines the

TABLE I.
DEPLOYMENT PARAMETERS

Parameter	Meaning of the parameters
n	The total number of network node NN who can participate in deployment in the ORN.
n_i	The number of NNs which need to be deployed in the i-th deployment path in traditional way.
$n_i^{'}$	The number of NNs which need to be deployed in the i-th deployment path in nested way.
p	Probability of successful deployment for a single NN.
t_o	The necessary preparing time for the first time MN need to deploy the entire transaction path.
t_a	The time it takes for the information packets transmission between MN and NN.
t_b	The time required for NN terminal to be ready to execute transactions.
α_i	The ratio of preparing time required for MN deploys i times and the required preparing time with the first deployment in traditional way, where $\alpha_{1=1}$, $\alpha_i = \frac{\sum_{k=1}^{i} n_k}{n_1}\alpha_1$
β_i	The ratio of preparing time required for MN deploys i times and the required preparing time with the first deployment in dynamic nested way, which $\beta_1 = 1$, $\beta_i = \frac{\sum_{k=1}^{i} n_k^{'}}{n_1^{'}}\beta_1$
T_i	The total time spent in traditional way
$T_i^{'}$	The total time spent in dynamic nested way

parameters in the TABLE.

The parameters are as follows:

(1) In the first deployment, the path are the same, so $n_1 = n_1^{'}$, $\alpha_1 = \beta_1$, $T_1 = T_1^{'}$;

(2) Because only the non-deployed nodes need to be redeployed in dynamic nested deployment, $n_i > n_i^{'}$, $T_i > T_i^{'}$;

The following Figure 3 and Figure 4 show the whole process that how MN deploys NN:

A. *The Advantage of Dynamic Nested Transaction Model*

1. The advantage of dynamic preparing subtransactions

Corollary 1: In case of single-function alternative set, the deployment can be successfully deployed in k wheels, the time spent in pre-prepared subtransaction is less than concurrent execution.

Proof: According to the prerequisites: $n = \lim_{k\to\infty}\sum_{i=1}^{k} n_i^{'}$, and functional alternative subtransactions present in k rounds, execution of $n_1^{'}$ NNs requires time t_0, so the

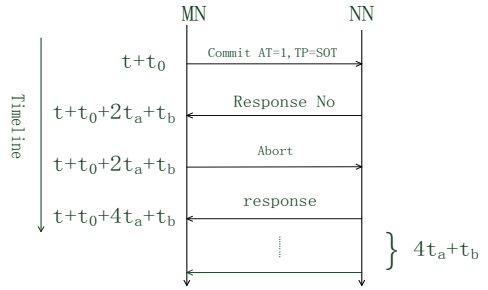

Figure 4. NN deployment failed

time required for n NNs is $\frac{n}{n_1^{'}}t_0$. Thus, the pre-prepared subtransaction time spent is:

$$T_1 = \frac{n}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n_1^{'} + (4t_a + t_b)\cdot n_2^{'} + \ldots + (4t_a + t_b)\cdot n_k^{'}$$
$$= \frac{n}{n_1^{'}}t_0 + (4t_a + t_b)\cdot \sum_{i=1}^{k} n_i^{'} \quad (1)$$

The concurrent implementation time spent is:

$$T_2 = \frac{n}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n \quad (2)$$

$$T_1 - T_2 = [\frac{n}{n_1^{'}}t_0 + (4t_a + t_b)\cdot \sum_{i=1}^{k} n_i^{'}] - [\frac{n}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n]$$
$$= (4t_a + t_b)\cdot (\sum_{i=1}^{k} n_i^{'} - n) \quad (3)$$

Because $n = \lim_{k\to\infty}\sum_{i=1}^{k} n_i^{'}$ and $\sum_{i=1}^{k} n_i^{'} < n$, we can get $T_1 < T_2$.

Corollary 2: In the case of successful deployment with a single function alternative set in k wheels, the time spent in dynamic preparing subtransaction is less than the pre-prepared transaction.

Proof: Dynamic preparing subtransaction time spent:

$$T_3 = \frac{n_1^{'}}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n_1^{'} + \frac{n_2^{'}}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n_2^{'} + \ldots + \frac{n_k^{'}}{n_1^{'}}t_0 + (4t_a + t_b)\cdot n_k^{'}$$
$$= \frac{\sum_{i=1}^{k} n_i^{'}}{n_1^{'}}t_0 + (4t_a + t_b)\cdot \sum_{i=1}^{k} n_i^{'} \quad (4)$$

$$T_3 - T_1 = [\frac{\sum_{i=1}^{k} n_i^{'}}{n_1^{'}} t_0 + (4t_a + t_b) \cdot \sum_{i=1}^{k} n_i^{'}] - [\frac{n}{n_1^{'}} t_0 + (4t_a + t_b) \cdot \sum_{i=1}^{k} n_i^{'}]$$
$$= (\frac{\sum_{i=1}^{k} n_i^{'}}{n_1^{'}} - \frac{n}{n_1^{'}}) \cdot t_0$$
$$= \frac{\sum_{i=1}^{k} n_i^{'} - n}{n_1^{'}} \cdot t_0 \quad (5)$$

Because $n = \lim_{k \to \infty} \sum_{i=1}^{k} n_i^{'}$, thus we can get: $T_3 < T_1$.

According to Corollary 1 and Corollary 2: $T_3 < T_1 < T_2$. Similarly, in multicast mode, Corollary 1 and Corollary 2 are also established. Therefore, above the conclusions, we can summarize the following theorem:

Theorem 1: In the case of successful deployment with an alternative set of single-function in k wheels, the time spent in dynamic preparing subtransaction is less than concurrent execution subtransaction and pre-prepared transaction.

2. The analysis of the advantages of dynamic nested deployment compared with traditional deployment

(1) In unicast and multicast deployment, the probability of successful deployment is the same.

1) The probability of using traditional deployment model to complete deployment within k rounds is:

$$p_1 = p^{n_1}$$
$$p_2 = p^{n_2}(1 - p^{n_1})$$
$$p_3 = p^{n_3}(1 - p^{n_1}) \cdot (1 - p^{n_2})$$
$$\vdots$$
$$p_{k-1} = p^{n_{k-1}} \prod_{i=1}^{k-2} (1 - p^{n_i})$$
$$p_k = p^{n_k} \prod_{i=1}^{k-1} (1 - p^{n_i}) \quad (6)$$

2) The probability of using dynamic nested deployment model to complete deployment within k rounds is:

$$p_1^{'} = p^{n_1^{'}}$$
$$p_2^{'} = p^{n_2^{'}}(1 - p^{n_1^{'}})$$
$$p_3^{'} = p^{n_3^{'}}(1 - p^{n_1^{'}})(1 - p^{n_2^{'}})$$
$$\vdots$$
$$p_{k-1}^{'} = p^{n_{k-1}^{'}} \cdot \prod_{i=1}^{k-2} (1 - p^{n_i^{'}})$$
$$p_k^{'} = p^{n_k^{'}} \prod_{i=1}^{k-1} (1 - p^{n_i^{'}}) \quad (7)$$

(2) As deployment of unicast mode, the communication between MN and NN strictly subjects to the number of nodes, while it is not in multicast mode. And thus the time spent is different.

1) Multicast mode

The deployment time it takes to be successful within k(k≥2) rounds in traditional model are as follows:

$$T_2 = \alpha_2 \cdot t_0 + 2(4t_a + t_b)$$
$$T_3 = \alpha_3 \cdot t_0 + 3(4t_a + t_b)$$
$$\vdots$$
$$T_k = \alpha_k \cdot t_0 + k(4t_a + t_b) \quad (8)$$

The deployment time it takes to be successful within k(k≥2) rounds in dynamic nested model are as follows:

$$T_2 = \alpha_2 \cdot t_0 + 2(4t_a + t_b)$$
$$T_3 = \alpha_3 \cdot t_0 + 3(4t_a + t_b)$$
$$\vdots$$
$$T_k = \alpha_k \cdot t_0 + k(4t_a + t_b) \quad (9)$$

2)Unicast mode

Time spent in the traditional deployment:

$$T_2 = \alpha_2 t_0 + (4t_a + t_b) \times (n_1 + n_2)$$
$$T_3 = \alpha_3 t_0 + (4t_a + t_b) \times (n_1 + n_2 + n_3)$$
$$\vdots$$
$$T_k = \alpha_k t_0 + (4t_a + t_b) \times \sum_{i=1}^{k} n_i \quad (10)$$

Time spent in the dynamic nested deployment:

$$T_2^{'} = \beta_2 t_0 + (4t_a + t_b) \times n_1^{'} + (4t_a + t_b) \times n_2^{'}$$
$$= \beta_2 t_0 + (4t_a + t_b) \times (n_1^{'} + n_2^{'});$$
$$T_3^{'} = \beta_3 t_0 + (4t_a + t_b) \times (n_1^{'} + n_2^{'} + n_3^{'}); \quad (11)$$
$$\vdots$$
$$T_k^{'} = \beta_k t_0 + (4t_a + t_b) \times \sum_{i=1}^{k} n_i^{'};$$

3. If the average deployment times for successful deployment of the transaction execution path in case of traditional deployment and dynamic nested deployment are respectively N_1 and N_2 in (12) and (13):

$$N_1 = \sum_{i=1}^{\infty} k \times p_i$$
$$= \lim_{k \to \infty} [1 \times p_1 + 2 \times p_2 + 3 \times p_3 + \ldots + (k-1) \times p_{k-1} + k \times p_k]$$
$$= \lim_{k \to \infty} [p^{n_1} + 2p^{n_2}(1 - p^{n_1}) + 3p^{n_3}(1 - p^{n_1})(1 - p^{n_2}) + \ldots$$
$$+ (k-1)p^{n_{k-1}} \prod_{i=1}^{k-2} (1 - p^{n_i}) + kp^{n_k} \prod_{i=1}^{k-1} (1 - p^{n_i})] \quad (12)$$

$$
\begin{aligned}
N_2 &= \sum_{i=1}^{\infty} k \times p_i^{'} \\
&= \lim_{k\to\infty}[p_1^{'} + 2\times p_2^{'} + 3\times p_3^{'} + \ldots + (k-1)\times p_{k-1}^{'} + k\times p_k^{'}] \\
&= \lim_{k\to\infty}[p^{n_1^{'}} + 2p^{n_2^{'}}\cdot(1-p^{n_1^{'}}) + 3p^{n_3^{'}}\cdot(1-p^{n_1^{'}})\cdot(1-p^{n_2^{'}}) + \ldots \\
&\quad + (k-1)p^{n_{k-1}^{'}}\cdot\prod_{i=1}^{k-2}(1-p^{n_i^{'}}) + kp^{n_k^{'}}\cdot\prod_{i=1}^{k-1}(1-p^{n_i^{'}})]
\end{aligned} \tag{13}
$$

We complete the digital experiment and the result is shown in Figure 5.

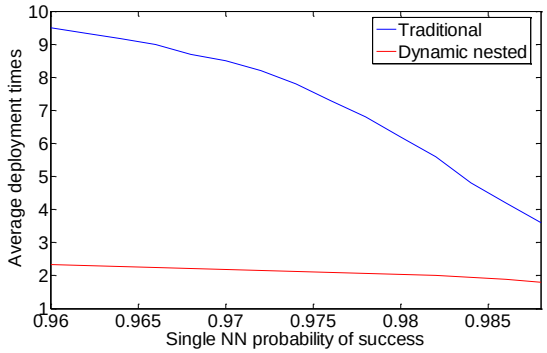

Figure 5. Average deployment times

The figure shows that in the case of successful deployment with a complete transaction execution path, the average deployment times of the dynamic nested deployment is less than the traditional deployment, and with the change of probability of success, there are big variations in times which is needed by traditional deployment, while the dynamic nested deployment is relatively flat. Thus, the dynamic nested deployment is better than the traditional deployment, and with the decreasing probability, its superiority is more obvious.

Theorem 2: In the case of successful deployment with a transaction execution path, the time spent in the dynamic nested deployment is less than the time spent in traditional deployment.

4. In order to make the time spent advantage of dynamic nested compared with traditional more visually, we have analyze deployment in two ways.

(1) Multicast deployment

In the traditional way, the expectation of successful deployment's time cost in two rounds is as follows:

$$
\begin{aligned}
E(T) &= T_1\cdot p_1 + T_2\cdot p_2 \\
&= T_1\cdot p^{n_1} + T_2\cdot(1-p^{n_1}) \\
&= T_1\cdot p^{n_1} + \frac{n_1+n_2}{n_1}T_1\cdot(1-p^{n_1})
\end{aligned} \tag{14}
$$

While in the dynamic nested transaction model, the expectation is:

$$
\begin{aligned}
E(T^{'}) &= T_1^{'}\cdot p_1^{'} + T_2^{'}\cdot p_2^{'} \\
&= T_1^{'}\cdot p^{n_1^{'}} + \frac{n_1^{'}+n_2^{'}}{n_1^{'}}T_1^{'}\cdot(1-p^{n_1^{'}})
\end{aligned} \tag{15}
$$

Comparing the efficiency of traditional deployment with dynamic nesting deployment by mathematical calculation:

$$
\begin{aligned}
E(T)-E(T^{'}) &= [T_1\cdot p^{n_1} + T_2\cdot(1-p^{n_1})] - [T_1^{'}\cdot p^{n_1^{'}} + T_2^{'}\cdot(1-p^{n_1^{'}})] \\
&\because n_1 = n_1^{'} \\
&\therefore p^{n_1} = p^{n_1^{'}} \\
&\therefore = p^{n_1}(T_1 - T_1^{'}) + (1-p^{n_1})\cdot(T_2 - T_2^{'})
\end{aligned} \tag{16}
$$

Because $T_1 = T_1^{'}$, $T_2 > T_2^{'}$, we can get : $E(T) > E(T^{'})$.

The expectation of time cost in traditional model within k rounds can be expressed as:

$$
\begin{aligned}
E(T) &= T_1\cdot p^{n_1} + T_2\cdot p^{n_2}(1-p^{n_1}) + T_3\cdot p^{n_3}\cdot(1-p^{n_1})\cdot(1-p^{n_2}) + \ldots \\
&\quad + T_{k-1}\cdot p^{n_{k-1}}\cdot\prod_{i=1}^{k-2}(1-p^{n_i}) + T_k\cdot p^{n_k}\cdot\prod_{i=1}^{k-1}(1-p^{n_i})
\end{aligned} \tag{17}
$$

While expectation of time cost in dynamic nested model within k rounds can be expressed as:

$$
\begin{aligned}
E(T^{'}) &= T_1^{'}\cdot p^{n_1^{'}} + T_2^{'}\cdot p^{n_2^{'}}\cdot(1-p^{n_1^{'}}) + T_3^{'}\cdot p^{n_3^{'}}\cdot(1-p^{n_1^{'}})(1-p^{n_2^{'}}) + \ldots \\
&\quad + T_{k-1}^{'}\cdot p^{n_{k-1}^{'}}\cdot\prod_{i=1}^{k-2}(1-p^{n_i^{'}}) + T_k^{'}\cdot p^{n_k^{'}}\cdot\prod_{i=1}^{k-1}(1-p^{n_i^{'}})
\end{aligned} \tag{18}
$$

We do digital simulation on deeply nested model and analyze the results, the experimental parameters are: t_0 =350ms, t_a =84.2ms, t_b =250ms, just as Figure 6 and Figure 7 show:

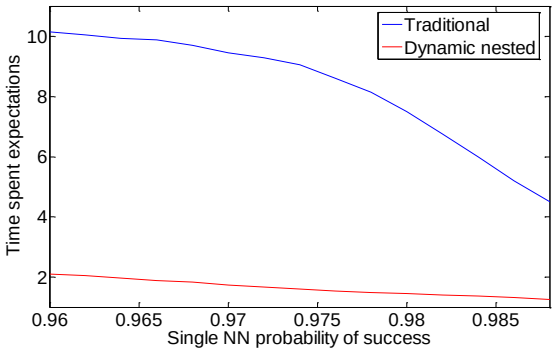

Figure 6. Multicast deployment

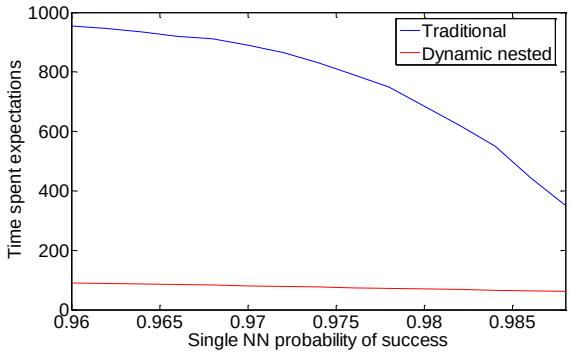

Figure 7. Unicast deployment

As we can see from Figure 6: With the probability of success increased, the times of deployment reduced, and thus the average deployment time spent in two ways of deployment also reduced. When the probability is relatively small, the superiority of the dynamic nested deployment compared with the traditional way is more obvious.

(2) Unicast deployment

In traditional way, the expectation of deployment's time cost in two rounds is:

$$E(T) = T_1 p_1 + T_2 p_2 = T_1 p^n + T_2 (1 - p^n) \quad (19)$$

While in the dynamic nested transaction model, the expectation is:

$$E(T') = T_1' \cdot p_1' + T_1' \cdot p_2' = T_1' p^{n_1'} + T_2' (1 - p^{n_1'}) \quad (20)$$

Comparing the efficiency of traditional deployment with dynamic nesting deployment by mathematical calculation:

$$\begin{aligned} E(T) - E(T') &= [T_1 p_1 + T_2 p_2] - [T_1' \cdot p_1' + T_1' \cdot p_2'] \\ &= [T_1 p^{n_1} + T_2 (1 - p^{n_1})] - [T_1' p^{n_1'} + T_2' (1 - p^{n_1'})] \\ &= p^{n_1} \times (T_1 - T_1') + (1 - p^{n_1}) \times (T_2 - T_2') \end{aligned} \quad (21)$$

Because $T_1 = T_1'$, $T_2 > T_2'$, we can get : $E(T) > E(T')$.

The expectation of time cost in traditional model within k rounds can be expressed as:

$$\begin{aligned} E(T) &= T_1 \cdot p_1 + T_2 \cdot p_2 + T_3 \cdot p_3 + \ldots + T_{k-1} \cdot p_{k-1} + T_k \cdot p_k \\ &= T_1 \cdot p^{n_1} + T_2 \cdot p^{n_2} (1 - p^{n_1}) + T_3 \cdot p^{n_3} \cdot (1 - p^{n_1})(1 - p^{n_2}) + \ldots \\ &\quad + T_{k-1} \cdot p^{n_{k-1}} \cdot \prod_{i=1}^{k-2} (1 - p^{n_i}) + T_k \cdot \prod_{i=1}^{k-1} (1 - p^{n_i}) \end{aligned} \quad (22)$$

While the expectation of time cost in dynamic nested model within k rounds can be expressed as:

$$\begin{aligned} E(T') &= T_1' \cdot p_1' + T_2' \cdot p_2' + T_3' \cdot p_3' + \ldots + T_{k-1}' \cdot p_{k-1}' + T_k' \cdot p_k' \\ &= T_1' \cdot p^{n_1'} + T_2' \cdot p^{n_2'} \cdot (1 - p^{n_1'}) + T_3' \cdot p^{n_3'} \cdot (1 - p^{n_1'}) \cdot (1 - p^{n_2'}) + \ldots \\ &\quad + T_{k-1}' \cdot p^{n_{k-1}'} \cdot \prod_{i=1}^{k-2} (1 - p^{n_i'}) + T_k' \cdot \prod_{i=1}^{k-1} (1 - p^{n_i'}) \end{aligned} \quad (23)$$

As we can see from Figure 7: When the probability is relatively small, the superiority of the dynamic nested deployment compared with the traditional way is more obvious.

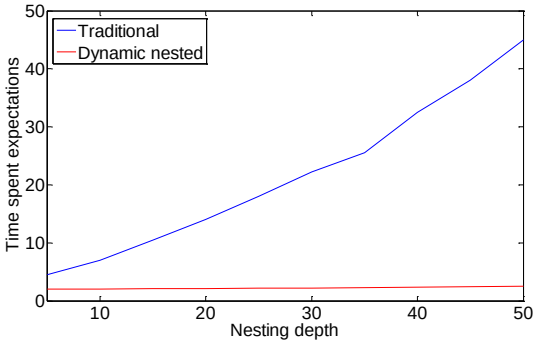

Figure 8. Multicast mode

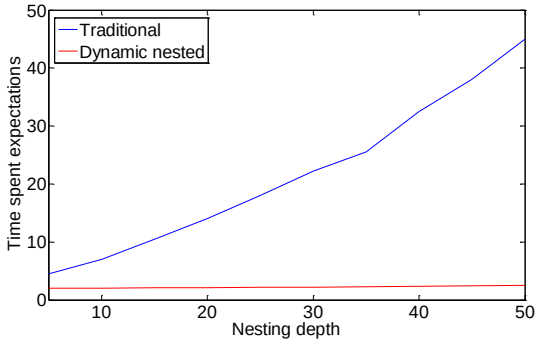

Figure 9. Unicast mode

B. *The Effect of Nesting Depth on the Two Transaction Models*

1. Multicast mode

The experimental parameters are: p=0.97, nesting depth of the initial value is 5, and every five increments, a total of 10 times' comparison. In different nesting depth, the time spent of the deployment in two ways is shown in Figure 8 and Figure 9.

From Figure 8, we can find that as nesting depth increases, the time spent by two ways of deployment increases, but the dynamic nested transactions' curve is relatively flat and its growth rate is relatively slow, while the traditional transactions' curve is almost linear speed growth. Thus, as the nesting depth increases, the dynamic nested transactions' performance is better than the traditional one.

2. Unicast mode

From Figure 9, we can find that in unicast mode the advantage of dynamic nested manner is more obvious. Because deployment time is influenced by impact of the number of nodes, and the number of nodes redeployed in dynamic nested manner is much smaller than the number of nodes redeployed in traditional deployment.

VI. CONCLUSIONS

In this paper, we studied the closed nested transaction model in the ORN. We did research on the advantage of the dynamic nested deployment which is compared with traditional deployment. Not only we came up with the rigorous mathematical proof, but also we validated through digital experiment to make the results more intuitive.

ACKNOWLEDGMENT

This work is partially supported by a grant from the National Basic Research Program of China (973 Program) under Grant No. 2012CB315902; the National Natural Science Foundation of China under Grant No. 61170215, 61102074; the Program for Zhejiang Leading Team of Science and Technology Innovation(No.2011R50010) and the Natural Science Foundation of Zhejiang (No.Y1111117).

REFERENCES

[1] WeiMing Wang. Forwarding and controlling Elements Separation(ForCES) technologies and applications [M].HangZhou: Zhejiang University press, 2010:25-38.
[2] Bochun Jia. Research and Implementation of open reconfigurable experimental platform [D];[Master's degree thesis] National Defense Science and Technology University, 2011.
[3] Luo Xu,Chaodong Ma,Yuanyuan Jia,Ennan Hao. Reconfigurable routing Development Environment. China Education Network, 2010.
[4] The national high technology research and development program (863 Program) information technology in the field of " a new generation of high trusted network" major project of 2009 year task application guide , http://www.most.gov.cn/tztg/200907/t20090705_71630.htm.
[5] Jia Yu, FangFang Deng. Survey of transaction processing technology [J]. Technology Square, 2008(8).
[6] Alia Bellabas, Samer Lahoud. Performance Evaluation of Efficient Solutions for the QoS Unicast Routing. Journal of Networks, Vol 7, No 1 (2012), 73-80.
[7] Seung-Jin Moon, Sanghoon Lee. A Reliable Nested transaction with Extension of Real-Time Characteristics. Reliable and Autonomous Computational Science, 2011.
[8] N. B. Lakhal, T. Kobayashi, H. Yokota. FENECIA: failure endurable nested-transaction based execution of composite Web services with incorporated state analysis. The VLDB Journal, 2009.
[9] Gang Li, Tao Wei, Wei Li, etc. Embedded database nested real-time transaction model[J]. Journal of Zhengzhou University of Light Industry, 2009(05).
[10] Yu Liu, Qiang Shen, Zhijun Zhao, Hui Tang.Proactive Location Service in Mobility Management. Journal of Networks, Vol 6, No 4 (2011), 670-677.
[11] Tony Hoare. Compensation Transactions. Advanced Lectures on Software Engineering, 2010: 21-40.
[12] Haitao Yang. A compensation cost analysis of service-aggregate transaction for DTNs clients. J.Parrallel Distrib.Comput, 2009, 69.
[13] GuoQiong Liao. Mobile distributed real-time database transaction and its correctness[D].[Doctoral thesis]. Huazhong University of Science and Technology, 2003.
[14] Luis A.Gama-Moreno, Matias Alvarado. Mobile nested transactions for nomadic teams. Expert Systems with Applications, 2004, 26:105-113.
[15] Sanjay Kumar Madria and S.N. Maheshwari and B. Chandra et al. An open and safe nested transaction model concurrency and recovery[C]. Journal of Systems and Software, vol.55, no.2, December, 2000, 151-165.
[16] Fekete,A.,Lynch,N.and Whiel,W.E. A serialization graph construction for nested transactions. In Proceedings of the International ACM Symposium, 1990:94-108.
[17] HaiBo Wu, MingWei Xu. ForCES architecture of router routing mechanism[J]. Journal of Tsinghua University: Natural Science Edition, 2008(01).

Jianping Qu is pursuing his master's degree in Communication and Information Systems at the school of Information and Electronic Engineering, Zhejiang Gongshang University, Zhejiang, China. His research interests include New network technology and Open Programmable Networks.

Ligang Dong, Ph.D., is currently the vice director of Institute of Network and Communication Engineering and vice dean of Information & Electronic Engineering in Zhejiang Gongshang University, China. His current research interests include: Open Architecture Network, Open Programmable Networks, ForCES, Distributed System, etc.

Lingjia Gui received her Master degree in Communication Information System in Zhejiang Gongshang University in 2011. Her main research interests include: Open Programmable Networks, Network Security, etc.

Weiming Wang, Ph.D., is currently the director of Institute of Network and Communication Engineering and dean of Information & Electronic Engineering in Zhejiang Gongshang University, China. His current research interests include: Information and Network Technology, Intelligent Information Processing, Linux and Embedded Systems, ForCES, Network Security, etc.