Research on the Cost of Open Nested Transaction in the Open Reconfigurable Network

Xiaolong Wang¹, Ligang Dong², Lingjia Gui³, Weiming Wang⁴, Julong Lan⁵

¹,²,³,⁴Zhejiang Gongshang University, Hangzhou310018, China
⁵National Digital Switching System Engineering & Technological Research Center, Zhengzhou450002, China

Email: wangxiaolong@pop.zjgsu.edu.cn, donglg@mail.zjgsu.edu.cn, guilingjia@pop.zjgsu.edu.cn, wmwang@zjsu.edu.cn, ljl@mail.ndsc.corn.cn

Abstract—Open reconfigurable network (ORN) is proposed to meet users’ diversified and dynamic requirements. Path deployment is a typical service in the ORN, and can be executed in the way of open nested transaction for high efficiency. In this transaction, each operation on network nodes (NNs) is a sub-transaction, which costs CPU resources. In this paper, first we build the cost model of the path deployment of three kinds of sub-transactions execution—prepare in advance, implement concurrently and prepare dynamically. Secondly, we analyze two main approaches of path deployment in the ORN—unicast and multicast, and we concluded that multicast costs more than unicast when we ignore the time. Finally, based on the comprehensive consideration of cost and time, we propose the optimal way to execute nested transaction in the ORN.

Index Terms—Open reconfigurable network, Nested transaction, Transaction costs

I. INTRODUCTION

Open reconfigurable network (ORN) defines mutual interfaces among network equipments, standardizes communication protocols and reconstructs the underlying resources of the network equipments on the same platform; it makes network equipments from different manufactures interoperable, compatible and flexibly to be functional reorganized. ORN is composed of integrated network management nodes (MNs) and a number of open reconfigurable routing and switching network nodes (NNs) [1]. Network nodes are managed by the management node, illustrated by the case of the path deployment, the management node controls resource reservation on network nodes in the ORN. So the execution of path deployment is a typical distributed transaction in the ORN.

A transaction is a set of predefined logical sequence of operations, and its implementation will complete a predefined function for the system. The characteristics of the ACID (Atomicity, Consistency, Isolation, and Durability) are proposed to ensure the correctness of the implementation [5]. In a distributed computing system, a transaction can access a number of different independent resources in different processes, or even on different computers. And we call this transaction a distributed transaction [2]. The traditional atomic transaction is composed of a flat sequence of operations; it has no internal structure and no strict compliance with the ACID characteristics. In the distributed network, two-phase commit protocol [3] is applied by the traditional atomic transaction model to ensure all distributed transactions be committed or rolled back together, so that the database stays in a consistent state. As long as one participant’s transaction fails, the entire transaction must be rolled back [6].

In the ORN, network nodes are controlled by one management node at least. A resource conflict occurs when a network node is ordered by different management nodes to run resource reservation at the same time. The conflict will greatly reduce the success rate of the transaction, and may happen repeatedly. Compared to the traditional atomic transaction model, the nested transaction model is proposed to improve the efficiency of transactions’ execution. The nested transaction model extends the traditional transaction model. In the nested transaction model, a parent transaction may contain multiple sub-transactions [8] [10], and some sub-transactions may have the same function. Corresponding author: Ligang Dong
transaction set composed of transactions with the same function is called a functional alternative set [11]. When to prepare a functional alternative set is depends, so in this paper, we divide the nested transaction model into the following three categories:

- **Prepare in advance**

  Sub-transactions have the same function are selected as a functional alternative set before the deployment. But only one of them is executed one time. Only if it fails, another transaction from the functional alternative set is selected to be executed.

- **Concurrent implementation**

  All sub-transactions from the functional alternative set are executed at the same time. But only one of them can be committed. This method increases the system overhead, but when the sub-transaction failure rate is fairly high, it saves re-deployment time.

- **Prepare dynamically**

  When the transaction is executed, all sub-transactions in the two-phase commit protocol return their first phase results (succeed or failure) after the first step. Then we dynamically generate the failure handling strategy by these results. Choose the optimal path for the deployment dynamically, rather than looking for a functional alternative sub-transaction set in advance. When a sub-transaction fails, look for another sub-transaction has the same function for a new execution.

In this paper, we provide the comparison on the costs of transactions executed in three ways: prepare in advance, implement concurrently and prepare dynamically. Then we calculate the costs of two different ways (unicast and multicast) for the path deployment in the ORN and make a comparison between them. Further more, based on the comprehensive consideration of cost and time, we propose the optimal way to execute nested transaction in the ORN.

### II. TRANSACTION MODEL

#### A. The Process of Path Deployment

The executed of path deployment is a typical open nested transaction in the ORN. The paper is focused on the costs of nested transaction in the ORN by studying path deployment.

To facilitate the expression, we allocate a separate determinate number (called ID) from \( \psi_n = \{1, 2, \ldots, n\} \) for each NN. \( NN_i \) denotes the network node (NN) with \( ID = i \). When the path deployment is executed, there are two kinds of NNs in the network. Some return successful messages showed in figure 1 and others return failure messages showed in figure 2.

In this paper, we provide the comparison on the costs of transactions executed in three ways: prepare in advance, implement concurrently and prepare dynamically. Then we calculate the costs of two different ways (unicast and multicast) for the path deployment in the ORN and make a comparison between them. Further more, based on the comprehensive consideration of cost and time, we propose the optimal way to execute nested transaction in the ORN.

<table>
<thead>
<tr>
<th>Sym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The total number of NN involved in the path deployment</td>
</tr>
<tr>
<td>( l_i )</td>
<td>The moment ( NN_i ) starts path deployment</td>
</tr>
<tr>
<td>( t_r(i) )</td>
<td>Time MN waits for ( NN_i )'s response</td>
</tr>
<tr>
<td>( t_s(i) )</td>
<td>Deployment deadline of ( NN_i )'s alternative node</td>
</tr>
<tr>
<td>( t_c(i) )</td>
<td>Compensation time for the committed node</td>
</tr>
<tr>
<td>( t_w(i) )</td>
<td>Time from ( NN_i ) receives the request to ( NN_i ) returns its result</td>
</tr>
<tr>
<td>( t_p(i) )</td>
<td>Time from ( l_i ) to MN decides to rollback</td>
</tr>
</tbody>
</table>

According to the nested transaction model, the MN sends a path deployment request to \( NN_i \) at time \( l_i \). If the MN receives a failure message from \( NN_i \) or doesn’t receive any response until \( t_r + t_r(i) \), it will try to find the alternative node of \( NN_i \). If it exists, it will replace \( NN_i \) to go on path deployment. If not, the transaction on \( NN_i \) will be rolled back. Before \( \max_{1 \leq i \leq \psi_n} (l_i + t_r(i)) \), the MN will make the final decision for the entire transaction. If it decides to abort, it will send abort messages to all NNs, and committed nodes need to
execute compensation transactions to come back to
the initial starting state.
To be fair, all NNs we mentioned in the paper have the
same \( t_i(i) \), \( t_j(i) \) and \( t_k(i) \).

**B. Definitions of Costs**
The TC (transaction-cost) in the ORN can be divided into
two types by different considerations.
If we only consider deployment costs
(1) For irreplaceable nodes, the T-C value is its own
cost of the CPU resources during the deployment,
denoted by \( c_i \);
(2) For the alternative node, the T-C value
\[
\begin{align*}
c_s(t) &= \begin{cases} 
c_i & \text{if } t-t_i < t_j(i) \\
c_i + c_{\alpha} & \text{if } t_r(i) < t-t_i < t_j(i) 
\end{cases}
\end{align*}
\]
When the deployment of a node is failed, the value is
the sum cost of itself and its alternative node.
If we consider both the deployment costs and
compensation costs
When the deployment on a node fails, committed
nodes and uncommitted nodes maybe both exist in
the system. Committed nodes need compensation and others
don’t. So the T-C value of a single network node \( NN_j \)
can be expressed as:
\[
c(t) = \begin{cases} 
0 & \text{executed unsuccessfully} \\
c_c(t) & \text{executed and uncommitted} \\
(c_c(t) + c_e(t)) & \text{committed}
\end{cases}
\]
If \( NN_j \) need no compensation, the T-C value of the
individual node \( NN_j \) is the cost of deployment; if it
needs compensation, the value is the sum of deployment
costs and compensation costs.
In the ORN, the deployment on each node is relatively
independent. And the successful probability of each
node is also relatively independent. We use
\( \rho_j \) (\( 0 < \rho_j < 1 \) | \( 1 \leq j \leq n \)) to express the success rate
on nodes \( NN_j \), so \( 1 - \rho_j \) is the failure probability. In the
process of path deployment, when the MN sends a
message to \( NN_j \) during the time \([t_j, t_j + t_j(j)]\), its
success rate can be expressed as
\[
\rho_{MN,j}(t_j, t_j + t_j(j)) = \rho_{MN+j}(t_j, t_j + t_j(j)) \cdot \rho_j.
\]
Among them, \( \rho_{MN+j}(t_j, t_j + t_j(j)) \) is the success rate of
messages transmission between MN and \( NN_j \). Because
the failure rate of message transmission between MN and
NN in the ORN is fairly small and can be ignored, in this
paper, we let \( \rho_{MN,j}(t_j, t_j + t_j(j)) = \rho_j \).
In the reconfigurable network, some nodes have
alternative nodes, so the success rate of deployment can
be divided into two categories:
(1) When \( NN_j \) has no alternative node: \( \rho_j = \rho_j \).
(2) When \( NN_j \) has an alternative node \( NN_k \),
then \( \rho_j = \rho_j + (1 - \rho_j) \cdot \rho_k \).
We use \( ETC(s) \) to indicate the ETC of scheduling
order \( S = \{t_1, k_1; t_2, k_2; \ldots; t_n, k_n\} \) and it is defined as
ETC of \( \{k_j \mid j \in \psi_n\} \), in other words, the sum ETC of
all nodes.

**III. Calculation and Comparison**

**A. Costs of Sub-transactions Execution**
When the path deployment begins, if all the nodes are
deployed at the same time, time spent the least; but it may
increase extra CPU costs and compensation costs. In this
paper, we analyze the costs of sub-transaction execution
by three different ways: prepare in advance, concurrent
implementation and prepare dynamically.
For the deployment sequence \( \{t_1, k_1; t_2, k_2; \ldots; t_n, k_n\} \),
we assume nodes with an even number have alternative
nodes, and nodes with odd numbers have no alternative
node.
If alternative nodes are executed, then we obtain:
(1) Prepare in advance
\[
ETC_1 = \sum_{k=2}^{n} \prod_{i=1}^{k-1} \rho_i (1 - \rho_i) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p(k)) \tag{1}
\]
(2) Concurrent implementation
\[
ETC_2 = \sum_{k=2}^{n} \prod_{i=1}^{k-1} \rho_i (1 - \rho_i) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p(k)) \tag{2}
\]
(3) Prepare dynamically
\[
ETC_3 = \sum_{k=2}^{n} \prod_{i=1}^{k-1} \rho_i (1 - \rho_i) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p(k)) \tag{3}
\]
Therefore, \( ETC_1 = ETC_2 = ETC_3 \).
If alternative nodes aren’t executed or don’t exist, then we
obtain:
(1) Prepare in advance
\[
ETC_1 = \sum_{k=2}^{n} \prod_{i=1}^{k-1} \rho_i (1 - \rho_i) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p(k)) \tag{4}
\]
(2) Concurrent implementation
When \( i = 2k - 1 \),
\[
ETC_2 (i) = \prod_{i=1}^{k} \rho_i (1 - \rho_i) \cdot \sum_{i=1}^{k} c_i (t_k + t_p(k)) \tag{5}
\]
When \( i = 2k \),
ETC' \( (i) = \prod_{v=1}^{k-1} \rho_v \cdot (1 - \rho_v) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p (k)) \)

When the number of failed node is even, we obtain:

\[
ETC' = \sum_{k=2}^{\infty} \prod_{v=1}^{k-1} \rho_v \cdot (1 - \rho_v) \cdot \left[ \left( \sum_{i=1}^{(k-1)/2} c_{2i-1} + \sum_{i=1}^{(k-1)/2} c_{2i} \right) \right] ^{1/2}
\]

(7)

When the number of failed node is even, we obtain:

\[
ETC' _2 = \sum_{k=2}^{\infty} \prod_{v=1}^{k-1} \rho_v \cdot (1 - \rho_v) \cdot \left[ \left( \sum_{i=1}^{k-1} c_i (t_k + t_p (k)) \right) \right] ^{1/2}
\]

(8)

3) Prepare dynamically

\[
ETC' _3 = \sum_{k=2}^{\infty} \prod_{v=1}^{k-1} \rho_v \cdot (1 - \rho_v) \cdot \sum_{i=1}^{k-1} c_i (t_k + t_p (k)) \]

(9)

From the above formulas, we can obtain the following results.

ETC' = ETC' _3

When \( i = 2k - 1 \),

ETC' \( (i) = ETC' \( (i) = ETC' _3 (i) \)

When \( i = 2k \),

ETC' \( (i) > ETC' \( (i) = ETC' _3 (i) \)

So ETC' > ETC' _3 = ETC' _2 _3

If alternative nodes aren’t executed or don’t exist, concurrent implementation perform all the alternative nodes, so it costs unnecessary CPU resources and results in a waste of CPU resources in the system. Based on the study of above-mentioned execution ways, we will focus on two main approaches of transaction scheduling.

B. Costs of Two Main Path Deployment Approaches

1) Unicast deployment

Definition 1: Let \( S = \{t_1, k_1, t_2, k_2, \ldots, t_n, k_n\} \) and it is a unicast deployment sequence. And if \( t_2 \geq t_1 + t_p (k_1), \ldots, t_n \geq t_{n-1} + t_p (k_{n-1}) \), particularly, when \( (\forall j \in \psi_{n-1}) t_{j+1} = t_j + t_p (k_j) \), then S is called the optimal unicast deployment sequence. Here, \( t_{j+1} = t_j + t_p (k_j) \) indicates that MN sends commit request to \( NN_{j+1} \) immediately after sending to \( NN_j \).

For the sake of simplicity, we use \( t_p (j) \) instead of \( t_p (k_j) \). In the process of unicast deployment, there are limitless possible sequences. The deployment sequence we talked about is arranged by the deployment speed on each node.

(1) ETC when only consider deployment costs

When the deployment on node \( k \) fails, it indicates that \( k-1 \) nodes in front are deployed successfully. The probability of this is \( \left( \prod_{v=1}^{k-1} \rho_v \right) (1 - \rho_v) \). Meanwhile, it occurs at \( t_k + t_p (k) \), and this moment is the deployment starting time of \( NN_k \)’s alternative node on condition that the transmission time is ignored. Thus, the ETC value of unicast deployment is:

\[
C_{\text{Unicast}} = \sum_{v=1}^{n} \left( \prod_{k=2}^{n} \rho_v \right) (1 - \rho_v) \sum_{i=1}^{k-1} c_i (t) \]

(10)

(2) ETC when consider both deployment costs and compensation costs

When the deployment on node \( k \) fails, the \( k-1 \) nodes in front are deployed successfully deployed and committed. So we need to consider the sum of deployment costs and compensation costs.

\[
ETC_{\text{Unicast}} = \sum_{k=2}^{n} \left( \prod_{v=1}^{k-1} \rho_v \right) (1 - \rho_v) \sum_{i=1}^{k-1} (c_i (t) + c_v (t)) \]

(11)

2) Multicast deployment

Definition 2: Let \( S = \{t_1, k_1, t_2, k_2, \ldots, t_n, k_n\} \) and it is a multicast deployment sequence. And if \( t_2 = t_1 + t_p (k_1), \ldots, t_n = t_{n-1} + t_p (k_{n-1}) \), particularly, when \( (\forall j \in \psi_{n-1}) t_{j+1} = t_j + t_p (k_j) \), then S is called the optimal multicast deployment sequence. Here, \( t_{j+1} = t_j + t_p (k_j) \) indicates that MN concurrently deploy all NN nodes at the same time. So in the multicast deployment:

\[
(\forall k) t_k = t_0 \]

and we use \( S_v = \{t_1, \ldots, t_n\} \) to indicate the multicast deployment set.

(1) ETC when only consider deployment costs

If the deployment on node \( k \) fails, it indicates that the deployments on the \( k-1 \) nodes in front are successful. All nodes in the multicast deployment are deployed at the same time, so the \( n-k \) nodes from \( (k+1, \ldots, n) \) have been deployed successfully, but not yet successfully committed. So we obtain:

\[
ETC(k) = \prod_{v=1}^{k-1} \rho_v \cdot (1 - \rho_v) \cdot \sum_{i=1}^{n} c_i (t) \]

(12)

\[
ETC_{\text{Multicast}} = \sum_{k=2}^{n} ETC(k)
\]

(13)

The above formula can become more intuitive. In this paper, we assume that \( \forall i \in (C1(i) \lor C2(i)) \).

C1(i): Node with its alternative node executed;

C2(i): Node without alternative node or don’t execute its alternative node;
In the process of the multicast deployment, \((\forall j \neq i) c_i(t_0 + t_j(f)) \equiv c_{ij}, \ c_i = c_i | c_i + c_u\).

Assume A = “node \(NN_j\) has been submitted”, B = “In addition to \(NN_j\), at least one node are committed unsuccessfully”, then we obtain:

\[
P(A) = \rho_i, \ P(B / A) = 1 - \prod_{v \neq j} \rho_v
\]

\[
P(A \cap B) = P(A) \cdot P(B / A) = \rho_i \cdot (1 - \prod_{v \neq j} \rho_v)
\]

Thus, the ETC value of \(NN_j\) can be expressed as:

\[
ETC(i) | Multicast = c_i \cdot \rho_i \cdot (1 - \prod_{v \neq j} \rho_v)
\]

(15)

The ETC of nodes \(NN_j\) can be expressed as:

\[
ETC(i) | Multicast = \sum_{i=1}^{n} P(B(i, k))
\]

(18)

\[
P(B(i, k)) = \sum_{j=1, j \neq i}^{k-1} \rho_j \cdot (1 - \rho_j) \cdot \rho_i
\]

(19)

Then we obtain the ETC value of the entire multicast deployment.

\[
ETC_Multicast = \sum_{i=1}^{n} ETC(i) | Multicast
\]

(17)

Similarly, we can simplify the expression.

\[
ETC_Multicast = \sum_{i=1}^{n} ETC(i) | Multicast
\]

\[
= \sum_{i=1}^{n} (c_i(t) + c_c(t)) \cdot \rho_i \cdot (1 - \prod_{v \neq j} \rho_v)
\]

(21)

The ETC when consider both deployment costs and compensation costs.

Assume \(A_i = " NN_j \) triggers compensation", \(B(i, k) = " NN_k \) is deployed unsuccessfully, therefore successfully deployed node \(NN_j\) need compensation”, so we get the following probability:

\[
P(A) = \sum_{k=1, k \neq i}^{n} P(B(i, k))
\]

\[
P(B(i, k)) = \sum_{j=1, j \neq i}^{k-1} \rho_j \cdot (1 - \rho_j) \cdot \rho_i
\]

(18)

(19)

Thus we can get the ETC of the entire multicast deployment.

\[
ETC_Multicast = \sum_{i=1}^{n} ETC(i) | Multicast
\]

(17)

\[
= \sum_{i=1}^{n} \sum_{k=1, k \neq i}^{n} (\sum_{j=1, j \neq i}^{k-1} \rho_j) \cdot (1 - \rho_j) \cdot \rho_i \cdot (c_i(t) + c_c(t))
\]

(21)

We let \(c_k = c_{ij}\) and then we obtain:

\[
C_{Unicast} = \sum_{i=1}^{n} \prod_{v=1}^{k} \rho_v \cdot \sum_{j=1}^{k-1} \rho_j \cdot \sum_{i=1}^{n} c_{ij}
\]

(24)

\[
C_{Multicast} = \sum_{i=1}^{n} c_i \cdot \rho_i \cdot \sum_{j=1}^{n} c_{ij} \cdot \prod_{v=1}^{n} \rho_v
\]

(25)

C. Comparison between Two Path Deployment Methods

We have proposed the unicast and multicast deployment, calculated their costs based on two aspects: consider deployment costs and consider both deployment costs and compensation costs. We compare them and obtain some results.

In the ORN, if only consider deployment costs, we can conclude that \(ETC_{Unicast} < ETC_{Multicast} \).

Proof:

(1) When \(\forall \ i \in C(2)\),

\(\forall \ i \in C(2)\), all nodes have no alternative node or don’t implement their alternative nodes. When all nodes fulfill these conditions and the node \(k\) fails: in unicast mode, the \(k - 1\) nodes in front are successfully deployed; in multicast mode, at least \(k - 1\) nodes have been deployed successfully, and \(c_{ij}(t) = c_{ij}\), all \(n\) nodes start their deployment. So we obtain:

\[
C_{Unicast} = \sum_{k=2}^{n} (\prod_{v=1}^{k} \rho_v)(1 - \rho_k) \cdot \sum_{i=1}^{n} c_{ij}
\]

(23)
$C_{Unicast} - C_{Multicast} = \sum_{k=1}^{n} \left( \prod_{i=1}^{k} \rho_i \right) c_s - \left( \prod_{i=1}^{n} \rho_i \right) \sum_{j=1}^{n} c_s$

\[= \sum_{i=1}^{n} \left( \prod_{i=1}^{k} \rho_i \right) c_s - \sum_{i=1}^{n} \left( \prod_{i=1}^{n} \rho_i \right) c_s \cdot \rho_i \]
\[= \sum_{i=1}^{n} \left( \prod_{i=1}^{k} \rho_i - \rho_i \right) c_s \]
\[< 0 \quad (26)\]

(2) When $\forall i \in C1(i)$, all alternative nodes needed are executed. When all nodes fulfill the condition, and the deployment of the node $k$ fails: In unicast mode, the $k-1$ node in front are deployed successfully and $c_i(t) = c_i + c_e$; in multicast mode, not all the $k-1$ nodes in front are deployed successfully, but they must have implement their alternative nodes, and all $n$ nodes start their deployment. So we obtain:

$C_{Unicast} = \sum_{k=2}^{n} \left( \prod_{i=1}^{k-1} \rho_i \right) (1 - \rho_k) \sum_{i=1}^{k-1} (c_i + c_e)$

$C_{Multicast} = \sum_{i=1}^{n} (c_i + c_e) \cdot \rho_i - \prod_{i=1}^{n} (c_i + c_e) \cdot \prod_{i=1}^{n} \rho_i$  

Therefore, $C_{Unicast} - C_{Multicast} < 0$.  

From (1) and (2), we can conclude that when only consider deployment costs, the cost of unicast deployment is less than multicast deployment.

When we consider both deployment costs and compensation costs in the ORN, we can conclude that $ECT_{Unicast} < ECT_{Multicast}$.

Proof:

In the ORN, when the deployment on a node fails, the number of successful nodes in unicast mode is larger than in multicast mode. And more nodes need to execute compensation. If every node has the same compensation costs, compensation costs in multicast mode is much more than in unicast mode. We have proven that deployment costs in multicast mode is large than in unicast mode, so it is proved.

D. Discussion About the Optimal Deployment Method in the ORN

From the research above, we find that the transaction cost in the multicast mode is more than in the unicast mode. But in the ORN, the deployment process is also limited by a deadline. In addition to the above-mentioned CPU costs, we should also consider the time it costs.

1) Unicast deployment

Analysis unicast deployment and we can get:

$t_j = t_0 + \sum_{i=1}^{n} t_p (i)$

$T_{Last} | Unicast = t_n + t_p (n) = t_0 + \sum_{i=1}^{n} t_p (i)$  

Among them, $T_{Last}$ is the moment when the entire transaction is decided to be committed or rolled back. In the unicast mode, the remaining alternative time of the network nodes $NN_j (T_{Left})$ is:

$T_{Left} (j) = t_j + t_s (j) - (t_0 + \sum_{i=1}^{n} t_p (i))$

\[= t_j (j) + t_j - t_0 - \sum_{i=1}^{n} t_p (i) \quad (30)\]

A negative $T_{Left} (j)$ means that there is no time to deploy alternative node. In the above formulas, there are two variables $t_s (j)$ and $t_j$, so that when $t_s (j) + t_j$ reaches the minimum, $T_{Left}$ is minimum, on the contrary, when $t_s (j) + t_j$ reaches the maximum, $T_{Left}$ is maximum.

2) Multicast deployment

From the analysis of multicast deployment, we can get:

$T_{Last} | Multicast = t_0 + \max_{i=1}^{n} t_p (i)$  

In the multicast mode, the remaining alternative time of the network node $NN_j$ is:

$T_{Left} (j) = t_j (j) + t_0 - (t_0 + \max_{i=1}^{n} t_p (i))$

\[= t_j (j) - \max_{i=1}^{n} t_p (i) \quad (32)\]

From the analysis of two deployment mode by time, we may conclude that:

(1) Multicast deployment can deploy faster than unicast to make a rollback or commit decision:

$T_{Last} | Multicast = t_0 + \max_{i=1}^{n} t_p (i)$

\[\leq t_0 + \sum_{i=1}^{n} t_p (i) = T_{Last} | Unicast \quad (33)\]

(2) Compare $T_{Left}$ values of the two deployment modes, then we obtain:
\[ T_{\text{Left}}(j) | \text{Unicast} - T_{\text{Left}}(j) | \text{Multicast} = t_j + t_0 - \sum_{i=1}^{n} t_p(i) - (t_j - \max t_p(i)) \]
\[ = t_j - t_0 + \max t_p(i) - \sum_{i=1}^{n} t_p(i) \]  
\[ = t_0 + \sum_{i=1}^{n} t_p(i) - t_0 + \max t_p(i) - \sum_{i=1}^{n} t_p(i) \]
\[ = \max_{i=1}^{n} t_p(i) - \sum_{i=1}^{n} t_p(i) \]

Typically, there is little differences among \( t_p(i) \) of all NNs, and \((\forall i) t_p(i) \geq \max_{i=1}^{n} t_p(i) / n \) is always valid, so \( T_{\text{Left}}(j) | \text{Unicast} - T_{\text{Left}}(j) | \text{Multicast} < 0 \). It means that multicast deployment has a longer \( T_{\text{Left}} \) than unicast, and deploy alternative network nodes in multicast mode is better when consider the deadline. In the ORN, there are two types of nodes, alternative nodes and non-alternative nodes, so we can combine the two deployment approaches to complete path deployment.

Assume: \( \exists C_0, C_1 \subseteq \psi_n \) and \( C_0 \cup C_1 = \psi_n \).

- **\( C_0 \): Nodes with alternative nodes**
- **\( C_1 \): Nodes without alternative node**

Under the above assumptions, the following deployment approach is the optimal by considering both deployment costs and compensation costs:

a) First, all nodes belong to \( C_1 \) execute multicast deployments;

b) When a) finished, complete all nodes belong to \( C_0 \) by executing unicast deployments.

We have obtained the conclusion that concurrent execution during the multicast deployment costs more CPU costs, so we choose unicast deployments. In order to improve the efficiency of the deployment, in other words, shorten the deployment time, we choose multicast deployments on the nodes with no alternative nodes.

## IV. Conclusion

The path deployment we study in the paper belongs to the nested transaction model. A single sub-transaction may be committed when it is completed. Compensation operations may have to be executed if it fails. In this paper, we study total costs of deployment and compensation, and prove merits of the two main deployment approaches—unicast deployment and multicast deployment. Based on these, we discuss the optimal deployment approach. In future studies, we may study the optimal depth of the nested transactions in the ORN and give a rigorous proof.

### ACKNOWLEDGMENT

The authors wish to thank School of Information & Electronic Engineering, Zhejiang Gongshang University. 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


Xiaolong Wang is a Master degree candidate whose specialty is Communication Information System at School of Information & Electronic Engineering, Zhejiang Gongshang University, Hangzhou, China. His main research interest is New Network Technology.

Ligang Dong received his Ph.D. degree in Computer Engineering in the National University of Singapore in 2003. He has been a professor and master tutor. He is currently the vice director of Institute of Network and Communication Engineering and vice dean of School of Information & Electronic Engineering in Zhejiang Gongshang University. 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 received his Ph.D. degree in Zhejiang University in 1992. He is currently the director of Institute of Network and Communication Engineering and dean of School of Information & Electronic Engineering in Zhejiang Gongshang University. His current research interests include: Information and Network Technology, Intelligent Information Processing, Linux and Embedded Systems, ForCES, Network Security, etc.