Available online at www.sciencedirect.com

ScienceDirect

J. Parallel Distrib. Comput. 67 (2007) 635–647

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

# A cooperative failure detection mechanism for overlay multicast

Mengkun Yang[a], Zongming Fei[b,*]

[a]Department of Computer Science, Eastern Kentucky University, Richmond, KY 40475, USA
[b]Department of Computer Science, University of Kentucky, 301 Rose Street, 2nd floor Lexington, KY 40506-0495, USA

Received 18 May 2005; received in revised form 5 March 2007; accepted 5 March 2007
Available online 14 March 2007

**Abstract**

Overlay multicast is widely accepted as an alternative to IP multicast for implementing group communications due to its easy deployment. One important issue to deal with is the node failures or ungraceful departures from the overlay multicast tree. Fast detection is a key to minimize the disruption of service to the affected nodes participating in the multicast session. In this paper, we propose a cooperative failure detection mechanism that can greatly reduce the failure detection time. We quantify three important measures, i.e., the expected detection time, the probability of false failure detection, and the overhead, and study the fundamental tradeoff among them in failure detection mechanisms. The analysis and simulations show that the proposed cooperative failure detection mechanism can significantly reduce the failure detection time while maintaining the probability of false positive at the same level, at the cost of slightly increased overhead.
© 2007 Elsevier Inc. All rights reserved.

*Keywords:* Failure detection; Overlay multicast; Heartbeat

## 1. Introduction

Point-to-multipoint communication is an important paradigm to achieve scalability of network communications, especially for those applications that target hundreds or thousands of receivers. IP multicast has been proposed as a network layer mechanism to implement the point-to-multipoint communication. However, IP multicast has not been widely deployed or enabled in the current Internet. One of the main reasons is that IP multicast routing protocols need network layer (or router) support and service providers are reluctant to enable IP multicast functions.

Overlay multicast [5] has been widely investigated as an alternative to IP multicast to implement group communications for its easy deployment. It is implemented at the end-hosts and does not require network layer (routers) support. It builds an overlay topology (usually a tree) among end-hosts participating in the multicast session, by using the unicast service provided by the substrate network. The duplication functions are implemented at end-hosts rather than routers. The overlay multicast has been used in live media streaming [10], bulk data transfer [14] and reliable group communication [2], etc. It has become a practical approach for implementing network applications that need to scale to a very large number of participants.

Management of overlay multicast trees faces a key problem. The non-leaf nodes in the tree are end-hosts, which are more likely to fail than routers and may leave the multicast tree voluntarily without informing other nodes. In these cases, all of its downstream nodes are partitioned from the multicast tree and cannot get the multicast data any more. It is important to recover from the partitioning quickly so that the disruption of service to those downstream nodes is minimized. The time to resume the data flow to those affected nodes is an important measure of the *responsiveness* of failure recovery mechanisms.

Before reconstructing the overlay multicast tree, we first have to detect the failure. When a node in overlay multicast fails or leaves the multicast session, other nodes should detect the event quickly. A departing node may leave the multicast tree gracefully, by sending a message to relevant nodes. The detection is not a problem in this case. However, it is not uncommon in overlay multicast that a node leaves the tree accidentally, such as in case of failures, or leaves voluntarily without informing other nodes about its status. We need a detection mechanism for affected nodes to reach the conclusion that the node is gone as soon as possible.

* Corresponding author.
*E-mail address:* fei@cs.uky.edu (Z. Fei).

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.03.002

In this paper, we propose a cooperative approach to node failure detection in overlay multicast to speed up the detection process. The basic idea is that all neighbor nodes interested in the well-being of a node will cooperate with each other. When any of them loses a heartbeat, it tells others in the group about this event. If a node cannot receive the heartbeats from the target node and also receives information from cooperating nodes that heartbeats are missing, it can reach the conclusion faster with higher confidence that the target node has failed. Through a formal analysis, we found that the cooperative scheme can achieve a shorter failure detection time and a smaller probability of false positive at the same time, with a small increase of overhead, compared with the basic heartbeat mechanism.

The contributions of the paper are as follows.

- Propose the idea of cooperative failure detection to speed up the detection process in overlay multicast.
- Quantify the expected detection time, the probability of false positive and the overhead of failure detection mechanisms, and analyze their tradeoffs.
- Propose solutions to several practical issues in implementing the cooperative failure detection scheme.
- Evaluate the proposed cooperative failure detection mechanism, by comparing with other mechanisms and testing various design parameters.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 proposes the cooperative failure detection mechanism. Section 4 gives an analytical study of different failure detection schemes. Section 5 discusses practical issues when implementing the cooperative scheme. Performance evaluations are presented in Section 6. We conclude the paper in Section 7.

## 2. Related work

Overlay multicast has been widely studied as a practical approach for group communications [5,2,7,11,4,13,8]. One of the major problems is how to deal with failures and recover the tree. The recovery process consists of two steps, *failure detection* and *tree reconstruction*. The tree reconstruction is the process that those orphaned nodes or subtrees find new parents to reconnect to the multicast tree. Several recent research papers have addressed the problem [10,6,3,16]. A centralized approach depends on a coordinator to find the locations in the tree for those affected nodes [10]. SpreadIt adopts a distributed approach in which disconnected nodes try to get help from their grandparents or the root of the tree [6,3]. A proactive approach pre-computes rescue plans before failures happen to reduce the recovery time [16]. They improve the performance of the tree reconstruction after failures have been detected.

In contrast, failure detection itself has not been widely studied in the context of overlay multicast. They are usually described as a part of the tree construction procedure [5,8,1,9]. In the end system multicast [5], each node sends *refresh* (heartbeat) messages to its neighbors. If a node does not receive refreshments from a neighbor for a long time, a probe is sent to this neighbor. The neighbor is considered to be dead if no response is returned. For those nodes it has not received refreshments for some time, but not long enough, it probabilistically probes them. In the intradomain overlays [9], each overlay node periodically sends *keepalive* (heartbeat) messages to its tree parent. Failure in receiving such a message makes the parent to probe the child. No response for the probe leads to the declaration of the child failure.

Both of them use the heartbeat mechanism to detect the failure. There are no cooperations among those nodes monitoring the same node. They also use probe to make sure that the target has failed, in order to reduce the probability of false positive. Notice that the final probe itself may also have a certain probability of false positive, depending on which probing method is used. While the probing can also be used in our systems, the cooperative scheme we proposed can reduce the time in which the monitor node detects the problem at the target node and increase the confidence of the conclusion. By choosing an appropriate threshold and forming a cooperative group, we can decrease the probability of false positive to such a low level that makes the probing unnecessary. We finish this section by describing an existing basic heartbeat scheme [5,9] and some analysis of its parameters.

### 2.1. Basic heartbeat scheme

The basic heartbeat scheme works as follows.

The node being monitored (called *target node*) sends a heartbeat message to each of its *monitor nodes* every $\mathcal{T}$ seconds. If a monitor node does not receive $k$ heartbeat messages in a row, it will derive that the target node has failed. The loss of a heartbeat message is also called a *heartbeat outage*.

Design parameters $k$ and $\mathcal{T}$ can be adjusted and determine the performance of the detection mechanism. Roughly, the failure of a node can be detected after $k\mathcal{T}$ seconds. A small $k$ will result in a short *detection time*. However, the value of $k$ cannot be too small. Notice that there are two reasons that the monitor node cannot receive heartbeats, either the node being monitored has failed, or the heartbeat is lost along the path. The monitor node is more likely to prematurely conclude that the node has failed, if $k$ is small. The probability of making a false conclusion in failure detection is called the *probability of false positive*. Increasing $k$ reduces the probability of false positive, but increases the detection time. There is a fundamental tradeoff between the expected detection time and the probability of false positive.

Another performance measure of interest is the *overhead*, i.e., the traffic generated by the heartbeat mechanism. It is determined in part by how often the heartbeat message is sent out to neighbors. A smaller heartbeat interval $\mathcal{T}$ leads to a shorter detection time, but the traffic generated will be heavier. This leads to the tradeoff between the expected detection time and the overhead of the detection mechanism.

Though these measures have been discussed in previous work, we have not seen a quantitative investigation of the tradeoff among them. In this paper, we will perform a formal analysis of the expected detection time, the probability of false positive and the overhead of various detection mechanisms by varying a small number of parameters (including $k$ and $\mathcal{T}$) and study the tradeoff among them.

A slightly different version of this scheme is that a monitor node will conclude that the target node has failed when $k$ out of $\ell$ ($k \leqslant \ell$) heartbeat messages have not been received. However, our goal here is to detect the failure of a node. After a node fails, it will not send any heartbeat messages to its monitor nodes. It is more reasonable for a node to conclude that the target have failed only after it cannot receive any heartbeat. Therefore, we will not discuss this alternative version any further.

Since each node independently makes its decision about the failure of other nodes based on its own observation on the heartbeat outages, we call this basic heartbeat scheme *non-cooperative* (NC) failure detection in our later discussion, in contrast to the cooperative scheme discussed next.

## 3. A cooperative failure detection mechanism

The node failure and its detection can be illustrated by an example in Fig. 1. When non-leaf node 5 fails, all the links (shown as dotted lines) between 5 and other nodes will be affected. All the downstream nodes, 8, 9, 10 and 15–22, are affected and experience data disruptions. Failure detection is the process that neighboring nodes (e.g., nodes 2, 8, 9 and 10) come to a conclusion that the target node being monitored (node 5) has failed. Failure detection schemes aim at achieving low detection time, low probability of false positive and low overhead. Note that every node in the overlay multicast tree can be a target node. The nodes participating in the failure detection of the target node are usually its *neighbors* connected with the target node, such as nodes 2, 8, 9 and 10 for target node 5, though it is possible that non-neighbor nodes are employed. In the discussion, we also use *monitor nodes* and *detecting nodes* interchangeably with *neighbors*.

We now propose our cooperative scheme for failure detection. The idea is to let the monitor nodes of a target node cooperate so that each node can reach the conclusion faster [17]. The monitor nodes for a target node constitute a cooperating group. For example, in Fig. 1, nodes 2, 8, 9 and 10 can be considered as a cooperating group. The goal is that in most cases, a monitor node can detect the failure of a target node within one heartbeat interval, with the help of the other nodes in the same group.

Similar to the non-cooperative case, every target node sends a heartbeat message to each of its monitor nodes every $\mathcal{T}$ seconds. When a monitor node does not receive the heartbeat message at the expected time, it will send a notification message to the other nodes in the cooperating group with regard to the same target node. When a monitor node receives the heartbeat message, no notification will be sent. So no extra traffic is generated when the target node works normally.

For a given target node, each monitor node maintains two counters in the cooperative scheme. One is the number of heartbeats lost, denoted as $N_{\mathrm{h}}$, and the other is the number of notifications received, denoted as $N_{\mathrm{n}}$. The cooperative failure detection algorithm can be described as Algorithm 1. Both counters $N_{\mathrm{h}}$ and $N_{\mathrm{n}}$ are initialized to 0. The algorithm then goes through a loop processing the (heartbeat, notification, and timer) events until it detects the failure of the target node. When it receives a heartbeat message from the target node, it resets both counters to 0 and restarts the timer. When it receives a notification message from the other nodes in the group, it increases counter $N_{\mathrm{n}}$ by 1 and check whether $(N_{\mathrm{h}} + N_{\mathrm{n}} \geqslant k) \wedge (N_{\mathrm{h}} > 0)$ is true. If yes, it concludes that the target node has failed. When the timer expires (indicating that it did not receive an expected heartbeat), it increases the counter $N_{\mathrm{h}}$ by 1, sends notifications to the other nodes in the group and checks whether $(N_{\mathrm{h}} + N_{\mathrm{n}} \geqslant k)$ is true. If yes, it concludes that the target node has failed;

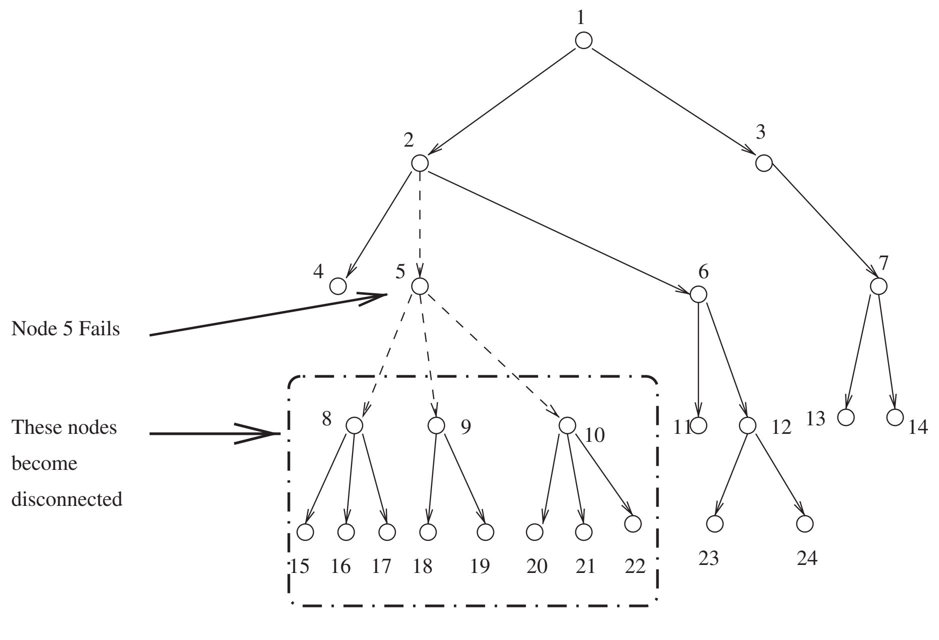


Fig. 1. An example of node failure.

**Algorithm 1**. Cooperative failure detection algorithm

```
1:  $N_h$ and $N_n$ are initialized to 0;
2:  Start the timer for the expected heartbeat arrival;
3:  **while** true **do**
4:      Wait for the next event;
5:      **if** the event is receiving a heartbeat message from the target node **then**
6:          $N_h = 0$; $N_n = 0$;
7:          Restart the timer;
8:      **else if** the event is receiving a notification **then**
9:          $N_n = N_n + 1$;
10:         **if** $(N_h + N_n \geqslant k) \wedge (N_h > 0)$ **then**
11:             Return (the target node has failed);
12:     **else if** the event is the expiration of the timer **then**
13:         $N_h = N_h + 1$;
14:         Send notifications to the other nodes in the group;
15:         **if** $(N_h + N_n \geqslant k)$ **then**
16:             Return (the target node has failed);
17:         Restart the timer;
```

Otherwise it restarts the timer. Note that $k$ is a threshold parameter that can be adjusted. It can be different from the $k$ in the basic heartbeat scheme. A monitor node reaches the conclusion that the target has failed only after it has lost at least one heartbeat from the target node.

## 4. Analytical study

To better understand the gains and cost of the failure detection approaches, we perform a formal analysis in this section. Specifically, we want to quantify the following performance measures:

(1) *Failure detection time*: It is the interval between the time a node fails and the time a monitor node reaches the conclusion that it has failed. It is an important performance measure that impacts the responsiveness of the failure recovery process.

(2) *Probability of false positive*: It is the probability that while a target node works fine, a monitor node reaches the conclusion that it has failed. This may happen when the heartbeat messages are lost. A wrong conclusion may waste network resources because it can initiate an unnecessary recovery process. We should keep the probability of false positive as low as possible.

(3) *Overhead*: It is the traffic generated for the failure detection purpose. Specifically, we calculate the average number of messages generated per unit time for a monitor node to be able to detect the failure of a target node. The overhead should be kept a very small portion of the total traffic so that it will not disrupt the normal multicast data flow.

We assume that the probability of message loss between any pair of nodes in the overlay multicast tree is $p$ $(0 \leqslant p \leqslant 1)$ and losses are independent. For clarity of analysis, we assume that the end-to-end path latency between two nodes is negligible and a monitor node knows a heartbeat is lost at exactly the same time when the heartbeat is supposed to be sent out. These paths include those not in the overlay tree and yet used in the cooperative failure detection. Every node in the overlay multicast tree fails with probability $q$ $(0 \leqslant q \leqslant 1)$. In the cooperative failure detection, we assume that the number of nodes in a cooperating group is $n$. When $n = 1$, there is no other node in the group. The other two important parameters are the threshold value $k$, which determines when a monitor node concludes that a target node has failed, and the heartbeat interval $\mathcal{T}$, which specifies how frequently the heartbeat messages are sent out.

### 4.1. Failure detection time

#### 4.1.1. Non-cooperative case

In the non-cooperative failure detection, a node can detect the failure of a target node if the number of heartbeats it has missed in a row exceeds the threshold $k$. Therefore, the failure detection time $T_{nc}$ is between $(k-1)\mathcal{T}$ and $k\mathcal{T}$, as shown in Fig. 2. Specifically, $T_{nc} \approx (k-1)\mathcal{T}$ if the target node fails immediately before the expected time at which a heartbeat should be sent out (point B); and $T_{nc} \approx k\mathcal{T}$ if the target node fails immediately after a heartbeat is sent out (point A). Assuming that the failure event is uniformly distributed in the interval (between points A and B), we can get the expected failure detection time for the non-cooperative approach as $E(T_{nc}) = \left(k - \frac{1}{2}\right)\mathcal{T}$.

Another non-cooperative approach is the probe mechanism, in which every monitor node periodically probes the target. If a monitor node has not received any response from the target in the last $k$ time intervals, it reaches the conclusion that the target has failed. Similar to the non-cooperative heartbeat method, the failure detection time $T_{probe}$ is between $(k-1)\mathcal{T}$ and $k\mathcal{T}$. If the target node fails immediately before the expected time at which a response should arrive, the detection time is $T_{probe} \approx (k-1)\mathcal{T}$. If the target node fails immediately after a response arrives, the detection time is $T_{probe} \approx k\mathcal{T}$. With the assumption that the failure event is uniformly distributed, we get the expected failure detection time for the probe approach $E(T_{probe}) = \left(k - \frac{1}{2}\right)\mathcal{T}$.

#### 4.1.2. Cooperative case

In the cooperative failure detection, the longest failure detection time is $T_{co} = k\mathcal{T}$. This is the case in which the target node fails immediately after a heartbeat is sent out and the detecting node gets no notifications of missing heartbeats from the other nodes in the group. Without help from other nodes, the failure detection is essentially the same as the non-cooperative approach. Hence, it can only determine that the target node has failed after it has missed $k$ consecutive heartbeats. Therefore, the time is $k\mathcal{T}$. Although this worst-case failure detection time is equal to that of the non-cooperative approach, it only happens with a very low probability. The shortest failure detection time is achieved when the target node fails immediately before a heartbeat is supposed to be sent out and the detecting node gets all notifications from all other nodes in the cooperating group. For every scheduled heartbeat, the detecting node gets $n-1$ notifications and one missing heartbeat. It takes $\lceil \frac{k}{n} \rceil$ missing heartbeats to get the total bigger than or equal to the threshold $k$. So the minimal detection time is $T_{co} = \left(\lceil \frac{k}{n} \rceil - 1\right) \times \mathcal{T}$.

We are interested in the expected detection time of the cooperative approach. Assume that the probability the detecting

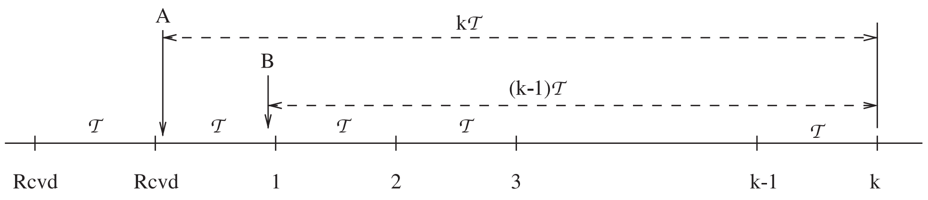


Fig. 2. Failure detection time by the non-cooperative approach.

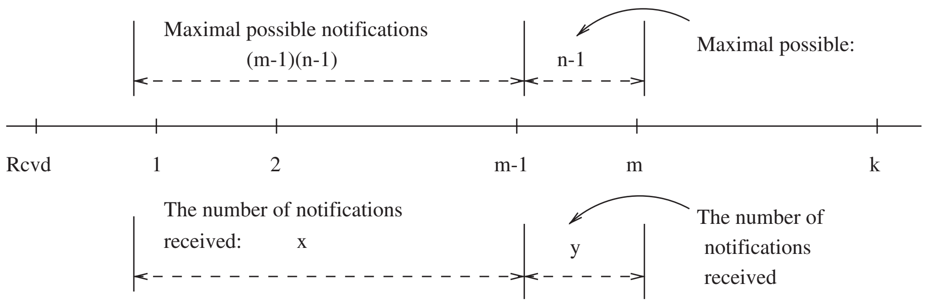


Fig. 3. Failure detection time by the cooperative approach.

node can reach the conclusion after $m$ ($\lceil \frac{k}{n} \rceil \leqslant m \leqslant k$) missing heartbeats is $g(m)$. The expected detection time will be

$$E(T_{\text{co}}) = \mathcal{T} \sum_{m=\lceil k/n \rceil}^{k} mg(m) - \frac{\mathcal{T}}{2}. \tag{1}$$

To get the probability $g(m)$, we explore all possible cases that a detecting node cannot make the conclusion in $m-1$ intervals, but can in $m$ intervals. Assume that the number of notifications received in the first $m-1$ intervals is $x$ and the number of notifications received in the $m$th interval is $y$, as shown in Fig. 3. Because the maximum number of notifications that can be received in one interval is $n-1$, we have $0 \leqslant x \leqslant (n-1)(m-1)$ and $0 \leqslant y \leqslant n-1$. The fact that the detecting node cannot make the conclusion in the first $m-1$ intervals implies $x+m-1 \leqslant k-1$, i.e., $x \leqslant k-m$. In order for the node to make the conclusion in the $m$th interval, we have $x+y+m \geqslant k$.

From $y \leqslant n-1$ and $x+y+m \geqslant k$, we get $x \geqslant k-m-(n-1)$. Therefore, $x$ can take values from $\max(0, k-m-(n-1))$ to $\min(k-m, (n-1)(m-1))$ and $y$ can take values from $k-m-x$ to $n-1$.

Because the target node has failed, all other nodes in a cooperating group will send a notification to the detecting node. Each notification will reach the detecting node with probability $1-p$. Under the independent loss assumption, the number of notifications reaching the detecting node in the first $m-1$ intervals follows the binomial distribution [12] with parameters $((n-1)(m-1), 1-p)$. The number of notifications reaching the detecting node in the $m$th interval follows the binomial distribution with parameters $(n-1, 1-p)$.

We use notation $P_{n,p}(i)$ to represent the probability of $i$ successes in the binomial distribution with parameters $(n, p)$. We know $P_{n,p}(i) = \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}$.

The probability of getting $x$ notifications in the first $m-1$ intervals is $P_{(n-1)(m-1),1-p}(x)$. The probability of getting $y$ notifications in the $m$th interval is $P_{n-1,1-p}(y)$. Therefore, if we let $\alpha = (n-1)(m-1)$ and $\beta = k-m-(n-1)$, the probability of reaching the conclusion in the $m$th interval is $g(m) = \sum_{x=\max(0,\beta)}^{\min(k-m,\alpha)} \left( P_{\alpha,1-p}(x) \sum_{y=k-m-x}^{n-1} P_{n-1,1-p}(y) \right)$. Combining this with formula (1), we get the expected detection time of the cooperative approach

$$E(T_{\text{co}}) = \mathcal{T} \sum_{m=\lceil k/n \rceil}^{k} m \sum_{x=\max(0,\beta)}^{\min(k-m,\alpha)} \times \left( P_{\alpha,1-p}(x) \sum_{y=k-m-x}^{n-1} P_{n-1,1-p}(y) \right) - \frac{\mathcal{T}}{2}. \tag{2}$$

## 4.2. *Probability of false positive*

### 4.2.1. *Non-cooperative case*

In the non-cooperative failure detection, a false failure detection at node $x$ is caused by the loss of $k$ consecutive heartbeats that the target node sends to $x$. Therefore, the probability of false positive is $F_{\text{nc}} = p^k$.

In the probe approach, both the probe message sent out by a monitor node and the response to the probe can be lost. A probe gets lost with probability $p$. The probability that a probe arrives at the target node, but the response to the probe is lost is $(1-p)p$. If either case happens, a monitor node will increase its counter for the missed responses and falsely determine that the target has failed when the counter reaches $k$. Therefore, the probability of false positive for the probe approach is $F_{\text{probe}} = (p+(1-p)p)^k = p^k(2-p)^k$.

### 4.2.2. *Cooperative case*

In the cooperative detection, both the consecutive loss of heartbeats sent to a monitor node $v$ itself and the notifications from cooperating nodes contribute to the false failure detection.

The probability that a notification from a cooperating node will be received by $v$ while the target node is still sending heartbeats is $\theta = p(1-p)$, which is the probability ($p$) that the heartbeat to that node is lost times the probability $(1-p)$ that the notification successfully reaches $v$.

The probability that $v$ concludes that the target node has failed after the target node sends out $m$ heartbeat messages is the probability that $v$ misses all $m$ heartbeat messages times the probability that an appropriate number of notifications reach $v$. Note that the number of notifications reaching $v$ in the first $m-1$ intervals follows the binomial distribution with parameters $(\alpha, \theta)$ and the number of notifications reaching $v$ in the $m$th interval follows the binomial distribution with parameters $(n-1, \theta)$. Following the similar reasoning as in deriving the expected time, the probability of false positive when node $v$ makes a decision that the target node has failed after $m$ ($\lceil k/n \rceil \leqslant m \leqslant k$) heartbeat messages is

$$f_{\mathrm{co}}(m) = p^m \sum_{x=\max(0,\beta)}^{\min(k-m,\alpha)} \left( P_{\alpha,\theta}(x) \sum_{y=k-m-x}^{n-1} P_{n-1,\theta}(y) \right), \quad (3)$$

where $\alpha = (n-1)(m-1)$ and $\beta = k - m - (n-1)$.

Therefore, the total probability of false positive in the cooperative case is

$$F_{\mathrm{co}} = \sum_{m=\lceil k/n \rceil}^{k} f_{\mathrm{co}}(m). \quad (4)$$

## 4.3. Overhead

### 4.3.1. Non-cooperative case

For a monitor node to be able to monitor the target, one heartbeat message is generated every $\mathcal{T}$ seconds, if the target node does not fail. The target node fails with probability $q$. Therefore, the overhead is $H_{\mathrm{nc}} = \frac{1-q}{\mathcal{T}}$.

The overhead in the probe approach arises from the probe messages and the responses. A monitor sends out one probe message per time interval $\mathcal{T}$. A response is sent back to the monitor node if the probe is not lost and the target works fine. Therefore, a response is sent out with probability $(1-p)(1-q)$. The overhead for the probe approach is $H_{\mathrm{probe}} = \frac{1}{\mathcal{T}} + \frac{1\times(1-q)(1-p)}{\mathcal{T}} = \frac{1-q}{\mathcal{T}} + \frac{1-p+pq}{\mathcal{T}}$.

### 4.3.2. Cooperative case

For the cooperative failure detection, in addition to the heartbeat message, we have notification messages transmitted between cooperating nodes. When the target node fails (with probability $q$), each monitor node will lead to $n-1$ notifications being transmitted. When the target node sends heartbeats normally (with probability $1-q$), each of $n-1$ cooperating nodes may miss the heartbeat message with probability $p$ and send a notification message to the monitor node. Therefore, the overhead for the cooperative case is

$$H_{\mathrm{co}} = \frac{q(n-1) + (1-q)(1+p(n-1))}{\mathcal{T}}. \quad (5)$$

Under the assumption that $p$ and $q$ are very small, the extra overhead of the cooperative approach is very small, compared with the non-cooperative case.

## 4.4. Numeral results and analysis

### 4.4.1. Non-cooperative approaches: heartbeat vs. probe

The analysis in Sections 4.1–4.3 shows that the non-cooperative heartbeat and probe approaches have the same expected failure detection time, under the assumption that the probe interval is the same as the heartbeat interval. However, the probability of false positive for the heartbeat approach is $p^k$, which is $(2-p)^k$ times lower than that for the probe approach. This is because both the probe from the monitor node and the response to the probe can be lost and thus the chance of false detection is increased in the probe approach. Moreover, the overhead for the heartbeat approach is $\frac{1-q}{\mathcal{T}}$, which is $\frac{1-p+pq}{\mathcal{T}}$ less than the overhead for the probe approach. This is due to the fact that both probes and responses put load on the network and the overhead is higher in the probe approach. Therefore, in the rest of the paper, when we study the performance of the cooperative failure detection, the non-cooperative heartbeat approach is used for comparison.

### 4.4.2. Non-cooperative vs. cooperative

We use “Coop” to represent the cooperative approach and “NonCoop” to represent the non-cooperative approach. The failure detection time is measured in the number of heartbeat intervals. For example, if the detection time is $1.5\mathcal{T}$, it corresponds to a $y$-axis value 1.5 in Figs. 4(a) and 5(b).

In Fig. 4, we show the numerical results comparing the non-cooperative approach and the cooperative approach. The number of nodes in a cooperative group ($n$) is either 2, 4 or 6. As the threshold ($k$) increases, the detection time increases in all schemes (Fig. 4(a)), but the cooperative approach increases much slower than the non-cooperative approach. For a given threshold value $k$, the cooperative approach can detect the failure in a significantly shorter time. For example, when $k = 4$, the cooperative approach with $n = 6$ only takes $0.5\mathcal{T}$, which is $\frac{1}{7}$ of the time ($3.5\mathcal{T}$) by the non-cooperative approach. This is because, in the cooperative approach, the failure detection is speeded up through the cooperation among the members in a cooperating group. However, in the non-cooperative approach, a monitor node cannot determine that the target has failed until it has missed $k$ heartbeats in a row.

Figs. 4(b) and (c) show the probability of false positive and the overhead, respectively. When using the same threshold value ($k$), the non-cooperative approach has a lower probability of false positive and lower overhead than the cooperative approach. The reason is that the non-cooperative approach does not incur extra messages for the purpose of detection in addition to the heartbeats. In contrast, the cooperative approach requires a monitor node send out notifications to the cooperating group, which leads to extra overhead. Moreover, when a monitor node misses a heartbeat due to message loss, its notifications to the group may result in false positives.

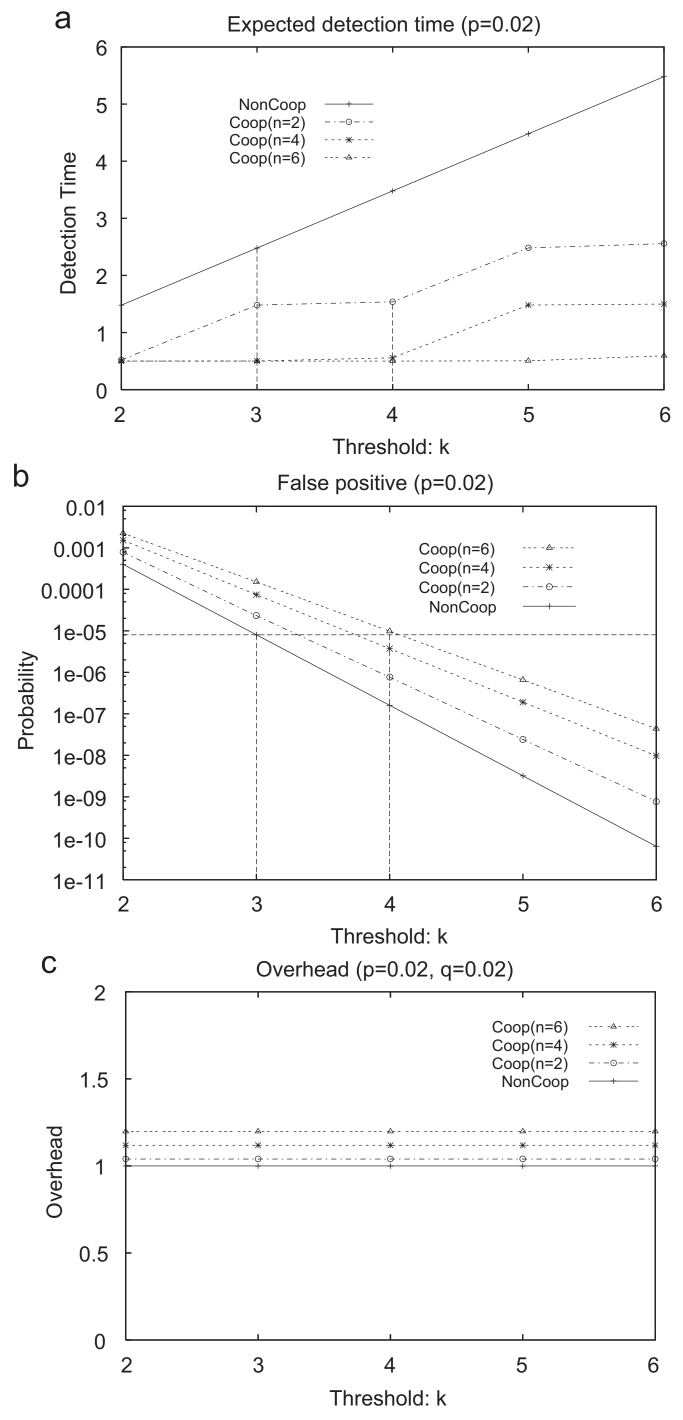


Fig. 4. Heartbeat mechanism: non-cooperative vs. cooperative: (a) detection time; (b) false positive; (c) overhead.

An interesting observation is that the cooperative approach can keep the probability of false positive the same, but achieve a shorter detection time. For example, we draw a horizontal line in Fig. 4(b). The cooperative scheme with $n = 6$ nodes in a group and the threshold $k$ equal to 4 has the same probability of false positive as the non-cooperative approach with threshold $k$ equal to 3, but it has a much shorter detection time ($0.5\mathcal{T}$) than the non-cooperative approach ($2.5\mathcal{T}$), as shown in Fig. 4(a). This means that the cooperative approach can decrease the detection time by 5-fold while maintaining the probability of false positive at the same level, at the cost of 20% more overhead.

Further, the cooperative scheme with $n = 4$ nodes in a group and the threshold equal to 4 achieves *both* a lower probability of false positive and a shorter detection time than the non-cooperative approach with $k = 3$. A more general observation from the plot is that given a non-cooperative scheme with some $k > 1$, we can always find a cooperative scheme with appropriate $k$ and $n$ such that it has a shorter detection time and a lower probability of false positive at the same time. In all cases, the extra overhead is always no bigger than 20%.

### *4.4.3. The number of monitor nodes in a cooperative group*

Next, we study the effect of the number of monitor nodes in a cooperative group on the performance of the detection mechanism. In Fig. 5(a), as the number of nodes in a group increases from 2 to 10, the detection time decreases from $2.5\mathcal{T}$ to $0.5\mathcal{T}$. Figs. 5(b) and (c) show that both the probability of false positive and the overhead increase when the number of nodes in a group increases. This is because when the target fails, a monitor node can get more notifications from a larger cooperating group and thus reach the conclusion on the target failure more quickly. However, a larger group also implies every monitor node send out more notifications when it misses a heartbeat, which leads to higher overhead. Furthermore, if the target works fine, a larger cooperating group increases the chances that during each heartbeat interval, at least one node in the group misses a heartbeat due to message loss. This can result in more false positives.

We make the observation that when the number of nodes in a group increases beyond threshold $k$, the detection time reaches the minimum value $\mathcal{T}/2$ and cannot be further decreased, while the probability of false positive and the overhead continue increasing. On one hand, this tells us that the cooperative scheme with $n \geqslant k$ achieves the design goal of having monitor nodes detect the failure of a target node within one heartbeat interval ($\mathcal{T}$) in most cases. On the other hand, this gives us a hint that the number of nodes in a group should not be too large. A value close to or a little bit bigger than $k$ is enough. In this way, when the target fails, a monitor node will receive about $k - 1$ notifications from the cooperating group. These notifications together with one heartbeat missed by itself are enough for it to reach the conclusion that the target has failed, according to Algorithm 1 in Section 3 because $N_h + N_n \geqslant k$.

## 5. Implementation issues

To implement the cooperative detection scheme, we need to deal with several practical issues. We design a timeout mechanism to detect the loss of heartbeats and develop a probabilistic notification technique for groups with a lot of members. In addition, we propose an asynchronous heartbeat mechanism to solve the implosion problem that may occur at monitor nodes.

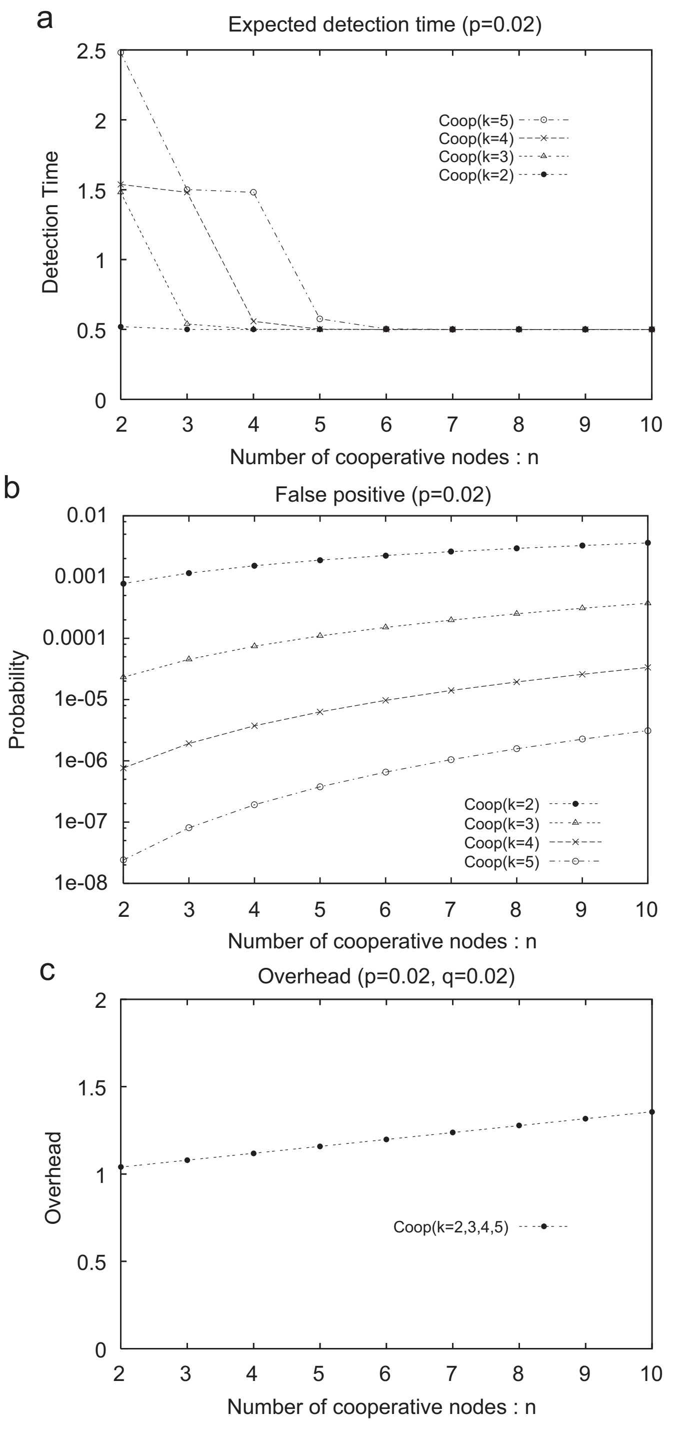


Fig. 5. Heartbeat mechanism: cooperative failure detection: (a) detection time; (b) false positive; (c) overhead.

**Algorithm 2**. Modified cooperative failure detection algorithm

```
1: $N_h$ and $N_n$ are initialized to 0;
2: Start the timer for the expected heartbeat arrival with value $2 * \mathcal{T}$;
3: while true do
4:   Wait for the next event;
5:   if the event is receiving a heartbeat message from the target node then
6:     $N_h = 0$; $N_n = 0$;
7:     Restart the timer with value $\mathcal{T} + \Delta$;
8:   else if the event is receiving a notification then
9:     $N_n = N_n + 1$;
10:    if $(N_h + N_n \geqslant k) \wedge (N_h > 0)$ then
11:      Return (the target node has failed);
12:  else if the event is the expiration of the timer then
13:    $N_h = N_h + 1$;
14:    Send notifications to the other nodes in the group;
15:    if $(N_h + N_n \geqslant k)$ then
16:      Return (the target node has failed);
17:    Restart the timer with value $\mathcal{T}$;
```

### 5.1. Detecting heartbeat losses

In the previous section, we assume that the latency between nodes is negligible and a monitor node knows a heartbeat is lost at exactly the same time when the heartbeat is supposed to be sent out. This simplifies the analysis, but in practice we need a timeout mechanism for nodes to detect the heartbeat losses.

We modify Algorithm 1 and add specific timeout value to the algorithm. Usually, the heartbeat interval $\mathcal{T}$ is much larger than the end-to-end latency. Initially, a detecting node can set the initial timeout value to a relatively large value, e.g., $2 * \mathcal{T}$. After receiving the first heartbeat message, it can set the timeout value to $\mathcal{T} + \Delta$, where $\Delta$ is added to accommodate the delay jitter between two nodes. Afterwards, when it receives a heartbeat message before the timer expires, it sets the timeout value to $\mathcal{T} + \Delta$; when the timer expires, the counter for heartbeat loss is increased by 1 and it sets the timeout value to $\mathcal{T}$. This is described in the modified cooperative failure detection algorithm in Algorithm 2.

The correctness of the mechanism can be explained as follows. If the detecting node does not receive any heartbeat, it will increase the lost heartbeat counter every $\mathcal{T}$ seconds until it is equal to $k$ and then conclude that the target node has failed. If the detecting node receives a heartbeat at $t$, the next heartbeat is supposed to arrive at $t + \mathcal{T} + \delta$, where $\delta$ is the difference between the current latency and the latency of the next heartbeat transmission, or the delay jitter. Since we set $\Delta$ big enough to accommodate the delay jitter $\delta$, the heartbeat will be able to arrive before the timer expires at $(t + \mathcal{T} + \Delta)$, if it is not lost. If the next heartbeat does get lost, the timer expires at $t + \mathcal{T} + \Delta$. Since we increase the lost heartbeat counter by 1 for every $\mathcal{T}$ seconds, it will get the correct count of lost heartbeats. Once a heartbeat is received, the whole process repeats again.

### 5.2. Detection groups and probabilistic notifications

We need to monitor the status of each node in the overlay multicast. Therefore, we need to form a monitor group for each node as a target node. From the analysis in the previous section, when the number of members in a group is greater than or equal to $k$, the detection time is minimized with an average time equal to about $\mathcal{T}/2$.

Naturally, the parent and children of a node are members of its detection group. We notice that the group formed this way may be too small. For example, leaf nodes only have a parent node, i.e., they have only one member in the group. The root node only has children and even an interior node in the overlay multicast tree may not have enough direct neighbors. In these cases, we can add siblings and other descendants to the group to make the number of members in a group greater than $k$.

The other extreme is that the group can be too large. The number of parent and children of some interior nodes can be much larger than $k$. As shown in Section 4.4.3, this does not help much to reduce the detection time further, but increases the traffic generated when a heartbeat is lost. We adopt a probabilistic approach in this case. We can let a node send notifications with probability $p_c$ $(0 \leqslant p_c \leqslant 1)$ if the number of members in a group is too large. Assume that the number of nodes in the group is $n$ $(n > k)$. We can set $p_c$ to about $\frac{k-1}{n-1}$, so that each monitor node can roughly receive $k-1$ notifications within one interval and detect the failure if there is one. By doing so, we reduce the overall traffic. We call this method the *probabilistic notification* scheme in the performance evaluation. In contrast, the original method will be called *non-probabilistic notification* scheme.

Another approach to reduce traffic is that we can use other received packets from the target node as an indication that the target node is still alive. This is especially useful when the target node is the parent of monitor nodes. This will prevent the monitor nodes send notifications to the other nodes in the same monitoring group. Even more, we can let the target node send the heartbeat messages piggybacked in the normal traffic, such as using a control flag field in the normal data packet to indicate this should be treated as a heartbeat from the target node. This can potentially significantly reduce the heartbeat traffic.

### 5.3. *Using asynchronous heartbeats*

In overlay multicast, a node usually cannot send the heartbeat messages to the monitor nodes at exactly the same time. The closest scenario is that after it sends a heartbeat message to one node, it immediately sends a heartbeat message to the next node. It has two problems. First, if the node is busy or cannot function for a short period of time scheduled for sending the heartbeat messages, its monitor nodes may conclude that it has failed and try to reorganize the tree. This premature decision may have a high cost. Second, when monitor nodes cooperate with each other to determine the status of a target node, each node will receive notification messages from the other nodes. This may cause the implosion problem because these notifications can arrive at almost the same time, if the latencies from these nodes are similar.

To solve the problem, we use an asynchronous mechanism for a node to send out the heartbeat messages. Instead of sending heartbeat messages at almost the same time, a node can schedule the heartbeat message to be sent to different nodes at different time. Specifically, we can distribute them evenly in each time interval $\mathcal{T}$. Suppose that there are $n$ members in a monitor group. The target node will send a heartbeat message to the $i$th $(0 \leqslant i \leqslant n-1)$ member at $i * \frac{\mathcal{T}}{n}$ after the start of each interval. The frequency of sending heartbeats is not changed, but we solve the possible implosion problem that may occur at the monitor nodes and reduce the possibility of the premature conclusion by the monitor nodes.

## 6. Performance evaluations

### 6.1. *Simulation setup*

In the simulation, we use GT-ITM [18] to generate 1600 node transit-stub topologies as the underlying network. One stub node is randomly chosen as the source and other end-hosts are randomly distributed in stub domains. The weights of the edges in the graph represent the network-layer link latencies, which range from 1 to 81 ms. The application-level distance (path latency) between two end-hosts is the sum of link latencies on the shortest path between them. In the generated topology, the path latency ranges from 1 to 220 ms with the average equal to 96 ms. The maximum number of neighbors that each node can have in the overlay multicast tree, called node degree, is uniformly distributed within range $[2, 2 \times d - 2]$, where $d$ is the average node degree. Note that $d$ potentially decides the average size of the monitoring groups in the multicast tree. So we control the average monitoring group size through varying $d$. In the simulations, $d = 5$ if not mentioned otherwise.

All experiments begin with a multicast tree established by the shortest path tree algorithm with the degree constraint taken into account. Initially, the total number of end-hosts in the tree is 160. Then end-hosts join and leave the tree dynamically, which is modeled as a Poisson process with leaving and joining rate $\lambda = \frac{1}{5}$ per second. Each experiment lasts for two hours.

We use the Gilbert model [15], which has often been used to reflect the bursty losses observed over the Internet, to simulate the packet loss on the network-layer links. In the simulations, the link loss rate $p_{\text{link}} = 1\%$ if not mentioned otherwise. This relatively high link loss rate biases against the cooperative approach. In an environment with less message loss, the cooperative approach can perform better than in the following simulation. The end-to-end path loss probability is $p = 1-(1-p_{\text{link}})^l$ if the path between a pair of end-hosts has $l$ links. The average number of links between neighbors in the generated multicast tree is about 6. In the experiments, the heartbeat interval is $\mathcal{T} = 15$ s.

Recall that we use "Coop" to represent the cooperative approach and "NonCoop" to represent the non-cooperative approach. Further, we use "Coop-nonprob" and "Coop-prob" denote the cooperative failure detection with the non-probabilistic and the probabilistic notification schemes, respectively.

### 6.2. *Failure detection time*

We study the failure detection time of different approaches and show the results in Figs. 6–9. As before, the failure detection time is measured in the number of heartbeat intervals.

Fig. 6 compares the average failure detection time. Fig. 7 depicts the cumulative distribution of the failure detection time when the threshold is $k = 4$. Curves "overall" plot the average detection time of all nodes in the multicast tree, including those

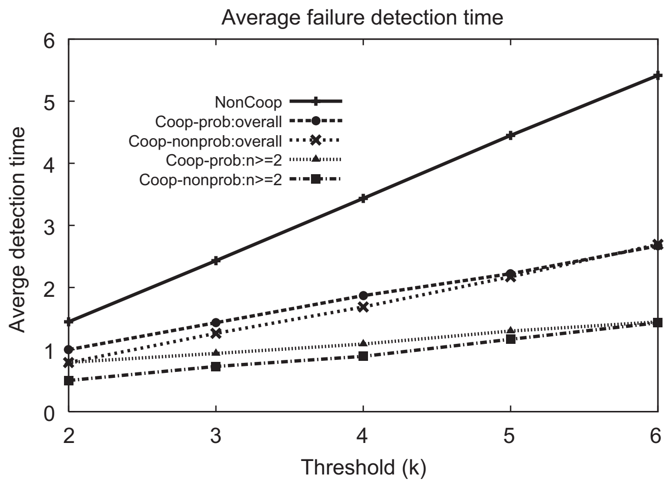


Fig. 6. Average failure detection time.

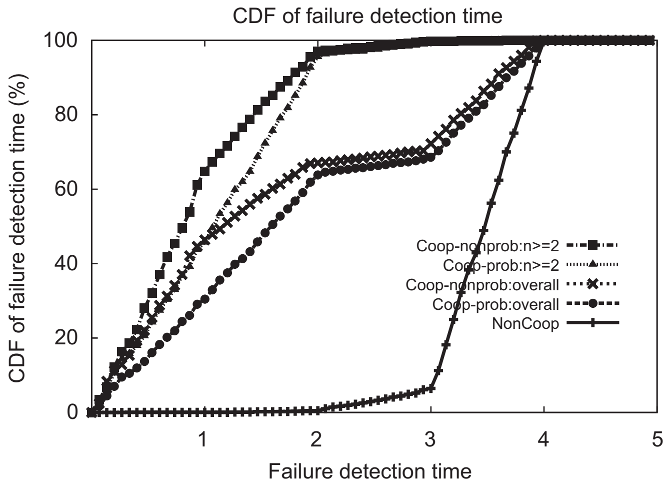


Fig. 7. Cumulative distribution of the failure detection time.

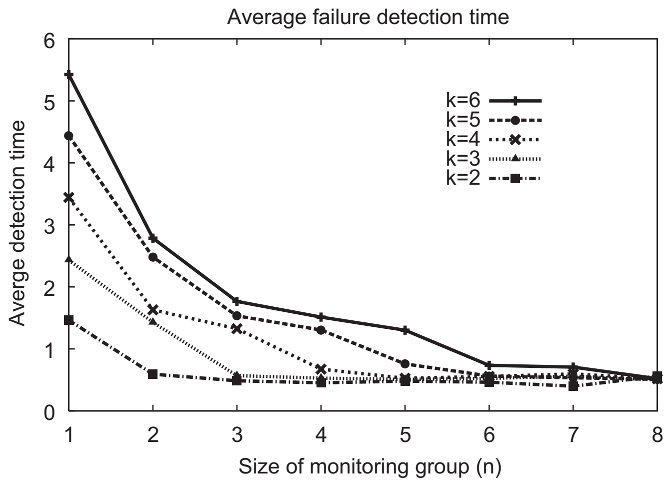


Fig. 8. Average failure detection time for the "Coop-nonprob" approach.

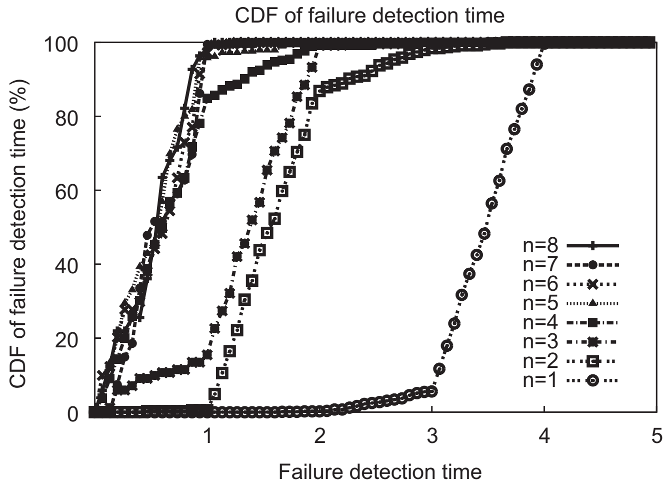


Fig. 9. CDF of the failure detection time for the "Coop-nonprob" approach.

having nobody to cooperative with, while curves "$n \geqslant 2$" only consider the nodes that have $n-1$ nodes to cooperate with to monitor the target nodes. We can see that the cooperative approaches, either using the non-probabilistic or the probabilistic notification, achieve much smaller average detection time, compared with the non-cooperative approach. For example, for the nodes in the monitoring groups with size $n \geqslant 2$, when $k = 4$, the average detection time is only 30% of the non-cooperative approach. Moreover, more than 55% of them detect the target node failures within just one interval and more than 95% of them detect the failures within two intervals. In contrast, using the non-cooperative approach, more than 95% of the failures are detected after three intervals. The detection time and its CDF of the "overall" curves are not as good as "$n \geqslant 2$" because the nodes having nobody to cooperate with take longer time to detect the failures. We vary the average node degree between 2.5 and 5 and find that about 70–85% of the nodes in the multicast tree belong to the monitoring groups with sizes no less than 2. This means that most of the tree nodes cooperate with somebody else and thus can benefit from the cooperative approach to detect the target node failures faster.

For further understanding the benefits on the failure detection time obtained by the cooperative approach, we present a more detailed data analysis in Figs. 8 and 9. Fig. 8 presents the average failure detection time for monitoring groups with different sizes. Fig. 9 plots their cumulative distribution of the failure detection time when the threshold is $k = 4$. Since nodes dynamically join and leave, or just move to another part of the tree if they infer that the parents fail, the size of the monitoring groups may vary over time. So it is hard to accurately define the size of the monitoring group which a monitor node belongs to. What we did is to use the monitoring group size at the time of making a detection decision. So $n = 1$ means that a monitor node has nobody to cooperate with. When the group size $n \geqslant k$, more than 85% of the failures are detected within one interval and the average detection time is approximately equal to $\frac{\mathcal{T}}{2}$, which is only $\frac{1}{2}$ and $\frac{1}{11}$ of the detection time for the group with

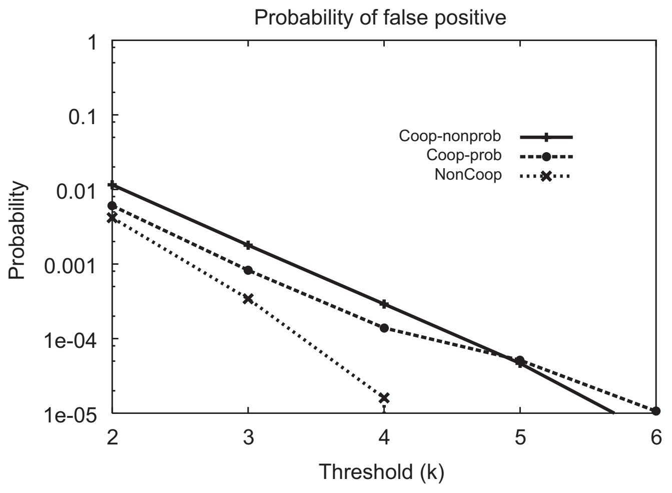


Fig. 10. Probability of false positive.

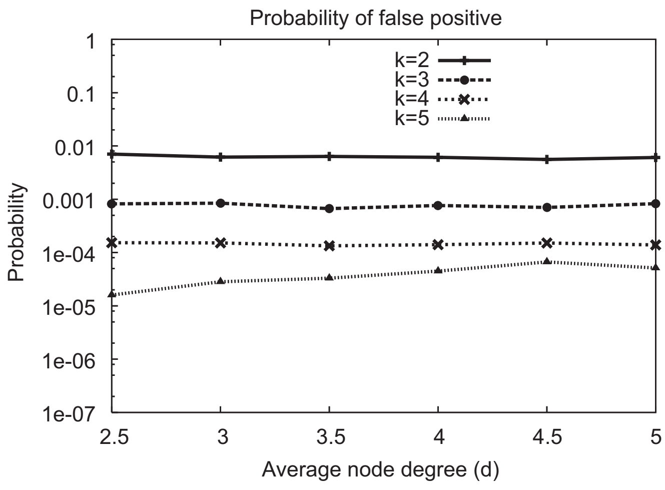


Fig. 11. Probability of false positive for the "Coop-prob" approach.

size $n = 1$ when $k = 2$ and 6, respectively. Another observation is, smaller groups, as long as its size $n > 1$, also help much in reducing the detection time. For these groups, more than 85% of the failures can be detected within two intervals, which is a great improvement over the non-cooperative approach where almost no failures can be detected within two intervals. Even having one cooperating monitor node (i.e., $n = 2$), no matter what the threshold $k$ is, can reduce the detection time by approximately 50%, compared with those nodes that have nobody to cooperate with at all.

Figs. 8 and 9 also explain why we observe the similar failure detection time when using different notification schemes (i.e., non-probabilistic and probabilistic) in the cooperative approach in Figs. 6 and 7. As the group size $n$ increases, the average detection time decreases. When the group size $n$ increases beyond threshold $k$, the detection time is approximately equal to $\frac{T}{2}$. Further increasing $n$ does not help much in improving the average detection time. We also find that the CDF curves of the detection time for all groups with sizes $n \geqslant k$ stay close to each other and a noticeable gap exists between them and those with $n < k$. So increasing the group size to $k$ effectively makes more nodes detect failures within a short time, but increasing the monitoring group size $n$ beyond $k$ does not make much difference. This is why the probabilistic notification achieves similar detection time as the non-probabilistic scheme.

### 6.3. *Probability of false positive*

Fig. 10 plots the probability of false positive. Given a threshold $k$, the cooperative approach has more false positives than the non-cooperative approach. However, we also note that using a larger threshold $k$ in the cooperative approach (e.g., $k = 4$) can result in the smaller probability of false positive and the smaller detection time than the non-cooperative approach with a smaller threshold (e.g., $k = 3$). This is similar to the observation made in the formal analysis. Also illustrated in Fig. 10 is that, compared with the non-probabilistic notification, the probabilistic scheme can reduce the probability of false positive for the cooperative approach when the threshold $k$ is less than the size of most monitoring groups.

Fig. 11 plots the probability of false positive for the cooperative detection with the probabilistic notification when the average node degree $d$ varies. A larger $d$ implies the larger average monitoring group size. Except for $k = 5$, the probability of false positive does not increase as $d$ increases. This is different from the analysis on the cooperative approach with the non-probabilistic notification in Section 4, showing that larger group size leads to more false positives. The reason for the difference is that with the probabilistic notification, when $n > k$, only about $k - 1$ (instead of $n - 1$ in the non-probabilistic scheme) cooperating nodes of a monitor node will send notifications to it. This also explains why the probabilistic notification leads to a smaller probability of false positive than the non-probabilistic scheme in Fig. 10.

### 6.4. *Overhead*

To evaluate the overhead of various detection approaches, we measure the number of packets sent per second for the detection purpose. Specifically, for the non-cooperative approach, the heartbeat packets are counted, while for the cooperative approach, both the heartbeats and the notification packets are included in the calculation. We define the *relative overhead* of the cooperative approach as the ratio of its overhead to that of the non-cooperative approach, under the same experiment configuration.

Fig. 12 depicts the relative overhead of the cooperative failure detection using two different notification schemes. We observe that using the non-probabilistic notification, the cooperative approach introduces about 20% more traffic than the non-cooperative approach. This allows us to reduce the average detection time for all tree nodes by 50% and for nodes having somebody to cooperate with by roughly 70–75%. Moreover, the overhead can be further reduced by adopting the probabilistic notification scheme. As demonstrated in the figure, its over-

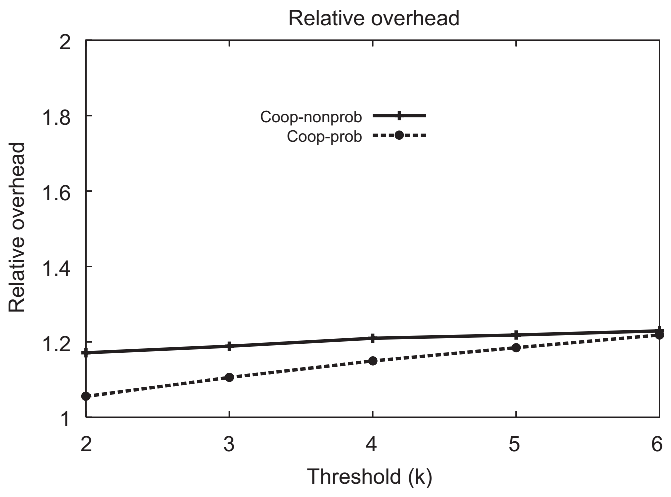


Fig. 12. Relative overhead of the cooperative approach.

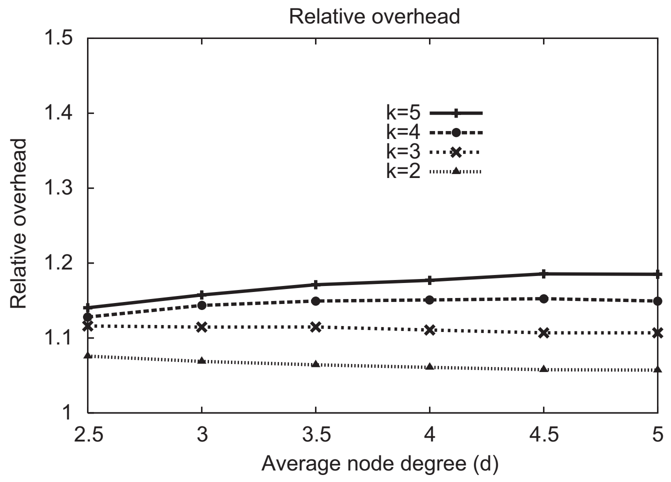


Fig. 13. Relative overhead of the "Coop-prob" approach.

head is always less than or equal to that of the non-probabilistic scheme. For example, with $k = 2$, the probabilistic notification only gives rise to 5% more traffic than the non-cooperative approach, in contrast to 17% in the non-probabilistic scheme.

Fig. 13 depicts the overhead as the average node degree varies. When using the probabilistic notification, when the threshold $k$ is small (i.e., $k = 2, 3$) relative to $d$, increasing $d$ does not lead to increased overhead. This is because, even though a monitoring group has more than $k$ members, the probabilistic notification makes a monitor node only send out roughly $k - 1$ rather than $n - 1$ notifications, when it misses a heartbeat. For larger $k$ (i.e., $k = 4, 5$), the overhead only increases slightly as the average group size increases. Together with the observation made from the previous study on the detection time, we find that increasing the monitoring group size effectively improves the detection time at the cost of very limited extra overhead.

## 7. Conclusion

Failure detection and recovery is a very important problem in overlay multicast. The effectiveness of the detection mechanism has direct impact on the service quality experienced by the receivers in the session. In this paper, we proposed a cooperative approach to speeding up the failure detection process. The salient feature of the approach is that a monitor node detects the failure of a target node not only by observing the missed heartbeats from the target node itself, but also from receiving notifications from other monitor nodes. Neighbors of a target node are usually interested in its well-being and can form a cooperating group to monitor the status of the target. They cooperate with each other by notifying group members if expected heartbeats are not received. This helps a node to detect the failure of the target faster with higher confidence. We gave a quantitative study of three important performance measures of detection mechanisms and analyzed the fundamental tradeoff among them. Both the quantitative analysis and performance evaluations show that, the cooperative approach allows failures to be detected within only one heartbeat interval, while the non-cooperative approach waits at least $k - 1$ intervals before reaching a conclusion that a node has failed. Meanwhile, the cooperative approach has comparable probability of false positive and usually incurs less than 20% extra overhead, compared with the non-cooperative detection. In addition, we also discussed the practical issues in implementing the detection mechanism. While the focus of this paper is on overlay multicast, the cooperative detection mechanism is also applicable to other overlay networks as well.

## Acknowledgments

This work was supported in part by the National Science Foundation under Grants CCR-0204304 and EIA-0101242, the U.S. Department of Treasury under Award #T0505060, and a grant from the Kentucky Science and Engineering Foundation as per Grant Agreement #KSEF-148-502-05-139 with the Kentucky Science and Technology Corporation.

## References

[1] D. Anderson, H. Balakrishnan, F. Kaashoek, R. Morris, Resilient overlay networks, in: Proceedings of ACM SOSP'01, 2001.
[2] F. Baccelli, A. Chaintreau, Z. Liu, A. Riabov, S. Sahu, Scalability of reliable group communication using overlays, in: Proceedings of the IEEE INFOCOM'04, 2004.
[3] M. Bawa, H. Deshpande, H. Garcia-Molina, Transience of peers and streaming media, in: Proceedings of HotNets 2002, 2002, Princeton, NJ.
[4] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: Proceedings of ACM Sigcomm'02, 2002, Pittsburgh, PA.
[5] Y.-H. Chu, S.G. Rao, S. Seshan, H. Zhang, A case for end system multicast, in: Proceedings of ACM SIGMETRICS'00, 2000, Santa Clara, CA.
[6] H. Deshpande, M. Bawa, H. Garcia-Molina, Streaming live media over a peer-to-peer network, Technical Report CS-2001-31, CS Department, Stanford University, 2001.
[7] P. Francis, Yoid: extending the Internet multicast architecture, ⟨http://www.icir.org/yoid/docs/yoidArch.ps⟩.

[8] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, J. James W. O'Toole, Overcast: reliable multicasting with an overlay network, in: Proceedings of the Fourth Symposium on Operating Systems Design and Implementation, 2000.

[9] C. Kommareddy, T. Guven, B. Bhattacharjee, R. La, M. Shayman, Intradomain overlays: architecture and applications, Technical Report UMIACS-TR 2003-70, University of Maryland, College Park, July 2003.

[10] V. Padmannabhan, H. Wang, P. Chou, Resilient peer-to-peer streaming, in: Proceedings of the 11th IEEE ICNP'03, 2003.

[11] D. Pendarakis, S. Shi, D. Verma, M. Waldvogel, ALMI: an application level multicast infrastructure, in: Third USENIX Symposium on Internet Technologies and Systems (USITS), 2001, pp. 49–60.

[12] S.M. Ross (Ed.), An Introduction to Probability Models, Academic Press, New York, 1997.

[13] A. Rowstron, A.-M. Kermarrec, P. Druschel, M. Castro, Scribe: the design of a large-scale event notification infrastructure, in: Proceedings of the Third International Workshop on Networked Group Communication (NGC), 2001.

[14] R. Sherwood, R. Braud, B. Bhattacharjee, Slurpie: a cooperative bulk data transfer protocol, in: Proceedings of the IEEE INFOCOM'04, 2004.

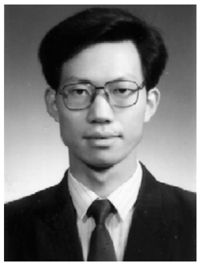
[15] M. Yajnik, S. Moon, J. Kurose, D. Towsley, Measurement and modelling of the temporal dependence in packet loss, in: Proceedings of INFOCOM'99, 1999, New York.

[16] M. Yang, Z. Fei, A proactive approach to reconstructing overlay multicast trees, in: Proceedings of the IEEE INFOCOM'04, 2004.

[17] M. Yang, Z. Fei, Cooperative failure detection in overlay multicast, in: Proceedings of Networking 2005, Waterloo, Canada, Lecture Notes in Computer Science, vol. 3462, Springer, Berlin, 2005, pp. 881–892.

[18] E.W. Zegura, K. Calvert, S. Bhattacharjee, How to model an internetwork, in: Proceedings of INFOCOM'96, 1996.

**Mengkun Yang** received the B.E. and M.E. degrees in computer science from Sichuan University in 1997 and 2000, respectively. She received the Ph.D. degree in computer science from the University of Kentucky, Lexington, KY in 2006. She is an Assistant Professor in the Department of Computer Science, Eastern Kentucky University, Richmond, KY. Her research interests include overlay networks, peer to peer systems, and multimedia systems.



**Zongming Fei** received the Ph.D. degree in computer science from Georgia Institute of Technology, Atlanta, GA, in 2000. He is an Associate Professor in the Department of Computer Science, University of Kentucky, Lexington, KY. His research interests include network protocols and architectures, overlay networks, mobile computing, and multimedia systems.