A Generic Resilient Multipath Routing Mechanism for Failure Prone Ad Hoc Networks

Mukil Kesavan, D. Amrit Swarup, K. Palanivel Andiappan,
{kmukil, amrit_swarup}@yahoo.com, palani.k@gmail.com

Department of Computer Science and Engineering,
College of Engineering, Anna University
Chennai, India

Abstract

In this paper, we propose a generic resilient multipath routing scheme with low overhead for mobile ad hoc networks with the aim of ensuring network throughput in both adversarial and node failure scenarios. Multiple paths of similar length between a given source node and a destination node are discovered on-demand. Then data is transmitted along these paths in a round-robin fashion with continuous path throughput monitoring. Paths which perform poorly in comparison with others are pruned from the active set of transmission paths. Our technique hinges on the fact that all security attacks or node failures in a mobile ad hoc network have the common symptoms of disruption of data communication which manifests itself in the form of reduced throughput.

I. Introduction

An ad hoc network is the cooperative engagement of a collection of mobile nodes without the required intervention of any centralized access point or existing infrastructure. Mobile nodes that are within each other's radio range communicate directly via wireless links, while those that are far apart rely on other nodes to relay messages as routers. Conventional single path routing protocols assume a friendly environment where mobile nodes co-operate to accomplish the task of data transmission. However, most conventional applications of mobile ad hoc networks still remain in the military domain that is prone to node failures and attacks. Also, many of the current approaches to provide fault tolerant operation are heavy weight and are designed to handle specific scenarios of attrition that do not cope well in the presence of other or new cases.

To address such issues, a light weight generic resilient routing mechanism is introduced in this paper with the design principle being toleration rather than prevention or detection. The aim is to provide network level reliability independent of transport and application layer security mechanisms. Our mechanism exploits the inherent redundancy of ad hoc network topology to achieve availability. Our multipath routing mechanism provides multiple paths of more or less similar length. The multiple paths form the basis of the resiliency architecture. Upon this, the path feasibility and pruning test is incorporated. This test ensures that data transmission takes place only along the

currently optimal paths from the set of available paths.

II. Related Work

Nahrstedt et al [1] proposes a new routing protocol called Best Effort Fault Tolerant Routing (BFTR) which builds upon the DSR protocol by adding fault tolerance by discovering multiple paths between the communicating nodes. In particular, they route using single paths initially and when the path becomes infeasible as per a two tailed test they drop the path and use another one. This protocol behaves more or less like a general single path protocol until a problem is discovered with the current path. We build upon this idea and augment it to capitalize on the better performance and load balancing provided by the usage of multiple paths.

Peter Pham et al [2] analyze the performance of on-demand single path protocols with that of multipath load-balancing protocols. Their work shows that in comparison with general single path routing protocol, multipath routing mechanism creates more overheads but provides better performance in a high density network when the route length is within an upper bound. They have also found out that overhead increases with the number of multiple paths used. In our mechanism we have focused on an idea proposed by Alvin Valera et al [3]; a shortest multipath routing where multiple paths of similar and short lengths are discovered. We have used this idea to take care of the “upper bound” and “reduced multiple paths” constraints of [2] which will provide better performance and reduced overhead as per the analysis of the same paper.

III. Assumptions

In order to provide resilient operation with the aim of tolerating faults or malicious activity, certain basic assumptions on the functionalities already provided is required. They are listed as below:

i. Source and destination nodes are assumed to be well behaved, which is a perfectly plausible proposition to start with. Routing between misbehaved nodes is not provided by the mechanism, which eventually results because of their misbehavior.

ii. A prior trust relationship between the communications ends i.e. source and destination is assumed as established to authenticate data transferred between them. Such trust relationship could be implemented using cryptography namely shared secret keys.

iii. A high-density network with at least one good route between nodes to facilitate meaningful communication. Now, a high-density network is assumed so that there can be found multiple paths between a given source and destination. The general deployment of ad hoc networks in real world follows reasonable density distribution of nodes across a finite limited space.

IV. Our Proposal

The operation of our mechanism can be broken down into three main phases.

Multipath Discovery

This resilient multipath routing algorithm works in an On-demand fashion. Firstly, when a node wants to send data to another node in the network, then it first checks its route cache for the availability of pre-discovered recent routes to the destination. If available then the route discovery process is bypassed and data transmission begins in a manner explained below. If there are no recent paths then the source node initiates a routes discovery process.

In the route discovery process there can be three types of nodes with respect to the role to be played. Firstly, a node can be a source node – it has to flood route request (RREQ) packets to all the adjacent nodes within radio range. A forward hop-count field fc is maintained for each packet transmitted. Fc is initialized to 0 at the source and is incremented every time it traverses a wireless link. Secondly, a node can be an intermediate node which receives a RREQ. Each such node already maintains a min (fc) field per pair of transmitting nodes which indicates the minimum path length traversed by a packet passing through this node. Min (fc) is initialized to a very high value. Now, the algorithm accepts to propagate an incoming packet if its fc is less than or equal to min (fc) and the necessary updates are performed. This ensures that a path from the source to this intermediate node is very close to the shortest available path in the network. Also, the node from which this route request came from is recorded in a predecessor list. If a node receiving RREQ is the required destination then it sends route reply (RREP) only if the fc of the incoming packet is less than or equal to its own min (fc). A corresponding backward hop-count hc initialized to 0 is tagged along with the RREP and the predecessor list is maintained appropriately. Thirdly, a node can be one, which receives a RREP. A mechanism similar to the forward propagation is followed and then if the RREP recipient is the source itself, then it records the route if it is at least less than or equal to in length to a previously discovered route. The route discovery process ensures that multiple available paths between source and destination with more or less similar path length are discovered.

```
route()
{
  if there are no routes in the cache for this destination then
  {
    discover new routes to this destination
  }

  While(true)
  {
    for each viable path
    {
      // transmit data and collect metrics. Also do simultaneous path monitoring and pruning.
    }
    if there are no viable paths left then
    {
      initiate new route discovery and store the new paths
    }
  }
}

//Discovering new routes

discoverRoutes()
{
  if sourceNode then
  {
    flood RREQ packets to neighbours with hopcount fc initialized to zero
  }
  if a RouteRequest from the Source to the Destination is received then
  {
```

```
      if this route is atleast as good as the
previous route in terms of hopcount fc then
      {
        Update the local hopcount field in the
node(self) and increment hopcount field of the
request
        Broadcast RREQ to adjacent nodes
      }
    }

 if  curentNode is the intended destination of a
RouteRequest then
    {
      if the hopcount fc of the packet is atleast as
good as that of the internal hopcount then
      {
        Prepare a RouteReply with hopcount hc
initialized to zero
        Transmit this to the previous node from
which the RouteRequest was obtained
        Update the internal hopcount of the
destination.
      }
   }


   if a RouteReply from the destination to an
intermediate node is received
   {
     if the hopcount hc of the packet is atleast as
good as that of the internal hopcount of the node
then
      {
        Increment the hopcount hc of the packet
and send the Reply to the node as indicated in
the packet
      }
   }
   if RouteReply from the destination to the
source node is received
   {
     Calculate the average hop count and RTT
combination of all the identified paths
      if the hopcount hc of the packet is atleast as
good as that of the internal hopcount of the
source then
      {
        Append the route to the set of routes;
      }
   }
   return the routes found;
}
```

Route Discovery algorithm

Packet Scheduling and metrics procurement

After finding multiple paths between a given transmitting node and a receiving node, packets are scheduled for transmission along the multiple paths in a strictly alternating manner (round robin). The entire packets payload is transmitted as solitary packets along each path i.e. one packet per path in cyclic order of transmission amongst paths. This is done primarily to load balancing and energy conservation which in turn helps in improving the resiliency of the network by reducing node failure due to wear out or resource unavailability.

Once a bunch of packets are sent via a path, the end-to-end performance of each path is evaluated through the use of network layer destination acknowledgments (ACK) and a decision on the subsequent viable usage of the path is made. Also, note that the use of network layer ACK packets are strictly for metrics procurement only and not for any reliability issues, as it will be taken care of by the upper TCP layer. The parameters taken into account for metrics calculation are end to end packet delivery ratio and packet delay.

Network layer acknowledgment packets are used to enable obtaining metrics such as:

GoodAcks - those acknowledgments that are received within a specified time limit which is a function of AvgRTT

NAcks - delayed or timed out acknowledgments

UsedCount - represents the number of times the given path has been used successfully

UpdateCount - the number times the given node has received an ACK from a given destination

RTT – round trip time between a given source and destination

AvgRTT - a running average reflecting the round trip time

PAvg - the average latency measure of a path from a given source to destination

NWAvg - the average latency measure of all the multiple paths from a given source to destination

BETA – simulation parameter dynamically varied to tune the mechanism toward better performance (0<BETA<1)

Note that dynamic mechanisms are used to evaluate paths, which reflect changing network conditions. Time-out is also used for the network layer ACK packets, which is roughly twice the AvgRTT. Upon the successful reception of an acknowledgment packet within time-out, GoodAcks is incremented and the corresponding failure event results in a unary increment of NAcks. Further, AvgRTT, PAvg, and NWAvg are calculated as follows:

AvgRTT = (AvgRTT * (UsedCount-1) + RTT) / UsedCount

PAvg = (1 – BETA) * GoodAcks + BETA * NAcks

NWAvg = (NWAvg * (UpdateCount -1) + PAvg) / UpdateCount

Path feasibility analysis and pruning

When a particular node wants to send data packets, it selects the path from a queue of multiple paths in cyclic fashion. PAvg plays an important role to decide whether a given path has to be selected or not. If the average latency of a particular path (PAvg) is greater than the overall latency of all the multiple paths connecting the same pair of source and destination (NWAvg), that path in consideration will be pruned. This pruned path is not deleted but maintained in the route cache. This is helpful when the NWAvg is continuously increasing and PAvg may fall below NWAvg at some point of time. This comparison is brought into effect only when the given path has already been used.

```
TransmitDataAndAnalyzePath(path P)
{
  Initialize the network average to zero;
  For each packet p_i sent along P
  {
    if acknowledgment for the packet arrives before a stipulated time then
      Increment it the good acknowledgments count good_ack;
    else
      Increment it the late acknowledgments count nack;
  }
  Calculate the path average as (1-β)*good_ack + β*nack where β is a run-time simulation parameter
  If path average is greater than the overall network average then
  {
    remove P from the set of viablePaths
    decrement the number of viablePaths
  }
  re-calculate the running network average;
}
```

Path feasibility analysis and pruning algorithm

V. Analysis

Security attacks and random node failures are handled by our mechanism in the same manner. We neither make an attempt to classify any disruption of network functionality nor try to waste precious resources in trying to identify the source of a problem. Both node failures and malicious node behavior have the common trait of manifesting themselves by decreasing end-to-end packet delivery ratio and increasing packet delay. Hence any path with a decreased performance will be pruned from the set of active transmission paths and the remaining paths between the communicating nodes are used to ensure data transfer. Our mechanism requires only end-to-end support primitives; no intermediate node support is needed. This makes our mechanism easily adoptable in any popular on-demand protocol such as DSR, AODV etc.

VI. Implementation and Results

The performance of the protocol was analyzed for various simulation parameters. The preliminary results are as shown below. The simulation was carried out in GloMoSim 2.03 with the following scenarios:

Parameters/values used

Node density (nodes, terrain)
50/ (100 x 100)
50/ (250 x 250)
50/ (500 x 500)
50/ (670 x 670)

No. of misbehaving nodes - 10%, 20%, 50%, 90%

BETA - 0.20, 0.50, 0.75, 0.80

Mobility models - Random waypoint (MAX SPEED: 10m/s and 1.5m/s)

Performance Characteristics

Throughput Analysis

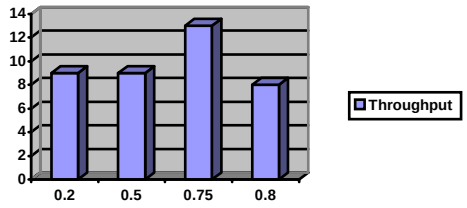

Fig 1: Throughput(y axis) vs. BETA(x axis)

The graph in Fig 1 depicts the results of throughput against the simulation Parameter BETA for a telnet client application. As can be seen, throughput peaks at BETA =0.75. This can be attributed to the fact that lower BETA values increases the number of paths pruned and therefore data is transmitted through a single path. Thus the throughput reflects the throughput of the possibly single path.

Pruned Count Analysis

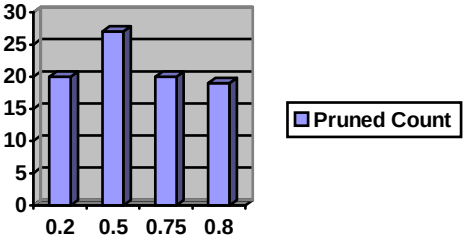

Fig 2: BETA(x axis) vs. Pruned Count(y axis)

The number of pruned paths peaks at BETA=0.5. For lower values of BETA, path count seems to increase than for higher values of BETA.

Node Density Analysis

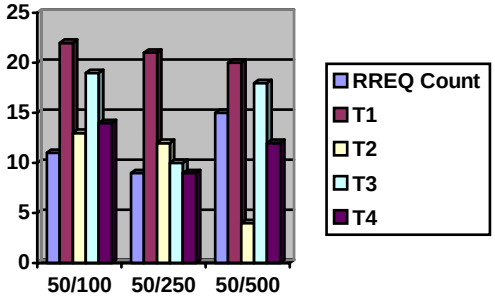

Fig 3: Throughput(x axis) vs. Node Density (y axis – no. of nodes/area^2)

(T1, T2, T3 and T4 are throughputs obtained for different applications)

As can be seen in the graph of Fig 3, throughput increases as the node density increases. This is attributed to the fact that with higher node density, there is more redundancy in the network and the mechanism exploits this redundancy by transmitting data along multiple feasible paths. Also the number of ROUTE REQUESTS decreases with increasing node density, since multiple paths are available and no additional paths are required than those already available.

VII. Future Work

We aim to further analyze the behavior and implications of our mechanism by incorporating it into many popular on-demand ad hoc routing protocols. Currently the mechanism takes into account end to end throughput and time metric for path evaluation; we plan to experiment with other parameters such as link reliability, node energy level etc.

VIII. Conclusion

In this paper we have presented a mechanism that uses multiple paths of similar short length to provide resilient routing service. Node failures and security breaches are managed resulting in improved reliability and graceful degradation of network functionality. We believe our work has duly explored the promising areas of ad hoc routing and fault tolerance research providing more insight into them for development of future applications.

References

[1] Yuan Xue and Klara Nahrstedt, Providing Fault-Tolerant Ad hoc Routing Service in Adversarial Environments Wireless Personal Communications 29: pages 367–388, 2004

[2] Peter P. Pham and Sylvie Perreau, Performance Analysis of Reactive Shortest Path and Multi-path Routing Mechanism with Load Balance, IEEE INFOCOM, 2003

[3] Alvin Valera, Winston K.G. Seah and SV Rao, Cooperative Packet Caching and Shortest Multipath Routing in Mobile Ad hoc Networks, IEEE INFOCOM, 2003

[4] Deepak Ganesan, Ramesh Govindan, Scott Shenker, Deborah Estrin, Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks, Mobile Computing and Communications Review, vol. 4, no. 5, October 2001

[5] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996