International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Proactive Calculation of K-Shortest Paths for Improving Performance of TCP in Ad-hoc Networks

Jyotsna Pandey
M.Tech. Scholar, TIT Bhopal
jyotsna2008@gmail.com

Sandip Nemade
Professor, TIT Bhopal
nemadesandip@yahoo.com

Vikas Gupta
HOD, Electronics, TIT Bhopal
vgtu24@yahoo.com

***Abstract* - The most dominant protocol in the Internet used is the Transmission Control Protocol (TCP). Most network application uses this protocol. The aim of this paper is to design TCP in such a way that it present end to end semantic for ad-hoc network. We will study TCP in such a way that how its design component will relate to the characteristics of ad-hoc network. Then we will deal with the different approaches to improve the performance of TCP protocol in ad-hoc network. We will present three approaches for improving performance. Then we will consider a solution to improve the performance by some proactive measures which will belong to one of the classes of the three approaches.**

***Keywords* – Ad-hoc networks, TCP, UDP, k-shortest path, ATP [9].**

I. INTRODUCTION

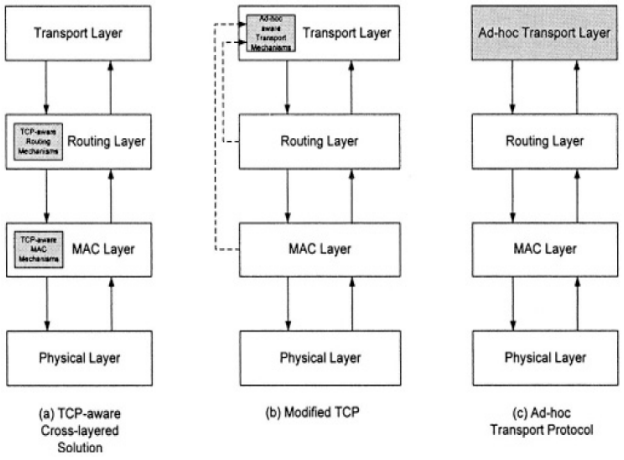
Transmission Control Protocol (TCP) provide connection oriented, reliable, end to end and in-sequence delivery of data as opposed to User Datagram Protocol (UDP), so TCP is quite complex. It also performs flow control, congestion control, recovery of data in case of loss and sequencing of data on the receiver side. TCP constitutes to about ninety percent of traffic over the internet and remaining traffic belongs to the protocol like UDP. Since overhead of headers and trailers associated with UDP is less so it is better to use UDP for real time application like transmission of multimedia over internet. So in the near future the share of traffic of UDP is about to increase. But still at present the TCP is leading the traffic share. So in this paper we will be focusing on designing of transport layer protocol for the ad-hoc network. The mobility in ad-hoc Networks will clearly lower down the performance of the ad-hoc network [14]. We can study the TCP protocol in both ways that is, in top down and bottom up fashion as discussed in [5]. Since TCP is the most dominant protocol so we are going to concentrate our focus on TCP. Initially we will study the TCP's design element relate to the characteristics of ad-hoc network protocol and we are going to check whether it is necessary or not. After studying reports we came to the conclusion that the redesign is must to improve the performance of TCP in ad-hoc network [5]. However we will also see whether solution is backward compatible or not.

II. THREE APPROACHES TO IMPROVE PERFORMANCE OF TCP IN AD-HOC NETWORK [19]

A. Modified TCP: In this approach minor modifications to TCP are made so that it can adapt it to the nature of ad-hoc Networks but at the same time TCP does not lose its fundamental characteristics as discussed in [1]

B. TCP aware Cross Layer Solutions: This solution works at the lower layer of TCP that hides the ad-hoc network's unique characteristics from TCP and thus results in minimal changes to TCP. This approach can be used along with above class as presented in [1]

C. Ad-hoc Transport Protocols: The last approach is to design a totally new protocol that is going to satisfy the necessity of ad-hoc network and it is not necessary that it is going to follow the characteristics of TCP protocol as discussed in [1].



III. RELATED WORK

A. Modified TCP

There is a protocol that falls under this category is Explicit Link Failure Notification (ELFN)[18]. It deals with the problems arising due to mobility. ELFN was proposed by Holland. ELFN takes the simple support from the network and lower layers to achieve its purpose. Most of the work in ELFN is done at transport layer with some new features along with the features of TCP protocol. The main purpose of ELFN to provide information about the route and link failures so that it can stop wastage of time for the failures in case of congestion and can avoid the degradation in the performance [1].

B. TCP-aware Cross-layered Solutions

There is an Atra Framework proposed by Anantharaman which falls under this category. This framework does not suggest any changes to TCP instead it is composed of mechanism at medium access control layers and in routing. The goals of Atra framework are as follows [1].

(i) The probability of route failures is to be minimized.
(ii) The route failures is to be predicted in advance

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

(iii) The route failure information should be conveyed in minimal possible time so that the latency is least.

C. Ad-hoc Transport Protocol

There is also a protocol which can be studied under the last approach. The advantage of this approach that it cab better suited to degree of characteristics of ad-hoc network. The name of the protocol is Ad-hoc Transport Protocol (ATP) proposed by Sundaresan[1][9]. The problem experienced by TCP over ad-hoc networks was overcome by this protocol and was implemented as different part of this protocol is a rate based transport protocol and its functionalities are in the form of three entities and these three entities are sender, intermediate nodes and ATP receiver [9]. The ATP sender is responsible for reliability, connection management, initial rate estimation and congestion control. The intermediate nodes are supposed to assist the sender by providing network feedback in terms of congestion control and initial rate estimation. The ATP receiver is responsible for assembling the feedback information provided by the intermediate nodes before sending the feedback to the ATP sender for rate and flow control and reliability [17].

IV. PROPOSED SOLUTION: PROACTIVELY CALCULATING K-SHORTEST PATHS

This is our proposed solution to improve performance and it falls under the category of second approach that is TCP-aware Cross-layered Solutions [11]. If a link failure occurs because of congestion or because of mobility. Then the packet has to be sent by some another shortest path. As soon as we detect some failure then in order to retransmit that packet through some another which has to be shortest one and has to be alive. Now if we are going to calculate another shortest path after the failure of the initial shortest path which is down at present, then it is going to take time and the latency to calculate new path increases which degrades the performance of TCP protocol [5]. In order to reduce this latency we can proactively calculate the k-shortest path at each node in advance which will reduce the time to calculate another path through which we can send the packets. Though there will be overhead in terms of computation power and memory requirement to store the k-shortest path but it will be fruitful if the network is highly congested [18]. Even if more than one shortest path is down, we are having backup of k-shortest path. Higher the value of k, better will be the performance and lower will be the latency but higher will be the cost. The k-shortest path can be calculated when the system is idle which is not going to affect the overall performance at the peak time. System may be regularly updated when the link goes up which is a mechanism that has been already implemented in TCP protocol we have to just pass the message in the k-shortest path link goes up. This approach performs well when the probability of link failure is very high and in case more than one shortest links are down. But if network is fit and fine the overhead is there for calculating the k-shortest path and storing them in the routers

V. ALGORITHM FOR CALCULATING K-SHORTEST PATH

The first of them is to find a set of K-shortest paths from source s to destination t, while the second requires finding all optimal paths, i.e., all paths from s to t with the same shortest length. In both cases, we assume non-negative weights. However, since we admit zero weights, the graph may contain cycles of zero length; such cycles cannot appear in our paths, thus we must be careful and make sure that the generated paths are simple (i.e., contain no cycles) as discussed in [2].

A. K-Shortest Paths

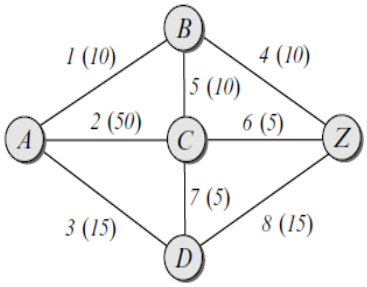
The K-shortest-paths problem is as follows: given graph G with non-negative link weights, source s and destination t, determine the shortest path, then the second shortest path, and so on, until the K-th shortest path [2]. The procedure solving this problem is described as Algorithm. The procedure returns the set of K-shortest paths K = {P1,P2, ...,PK}.

For K = 1 the algorithm finds the single shortest path and stops. Otherwise, in order to find each of the next shortest paths beyond the first one, a fairly sophisticated procedure is applied. The procedure makes use of an auxiliary set S, composed of pairs of the form (Q, v), where Q is a path from s to t and v Qˆ , which is then used to adjust the set X of candidate shortest paths. In the algorithm, _ denotes the operator opposite to the concatenation operator. Also, subQ(v,w), where v,w Qˆ , denotes the sub-path of Q from node v to node w (provided v appears earlier than w). The procedure uses the shortest path P added to set K in the previous iteration to adjust the set of candidate paths X. Finally, after doing this, it finds the shortest path in this set (using function shortest(X)), and adds it, as the consecutive shortest path, to set K. Set X is adjusted in the following way. First we use function GetDeviation Vertex (S,P) to find the unique pair of the form (P, w) in set S and the corresponding deviation vertexw associated with pathP. Then we consider all the consecutive vertices (except destination t) in the sub-path subP(w, t) (they are called deviation vertices). For each such vertex v we find the shortest path from source v to destination t, referred to as the deviation path, and concatenate this path with the sub-path of P, subP(s, v), starting in s and terminating in v, to form pathQfrom s to t. PathQis then added to set X. In order to assure that pathQ has not already been generated before running the Dijkstra algorithm, we need to modify the graph, by removing certain nodes and edges, using function DisableV erticesAndEdges. This function removes all vertices forming the sequence subP(s, v) _ v from graph G, together with all edges incident to the deleted vertices. Note that this also assures that the newly constructed pathQis simple. Additionally, for each previously found shortest pathP_ K{P} with the property: subP(s, v) = subP_ (s, v), we remove its edge outgoing from vertex v towards t. After finding path Q for each deviation vertex v we add pair (Q, v) to set S. This algorithm is presented in [2].

International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 5, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

```
procedure K-SP(G, s, t,K)
k := 1;
P := SPD(G, s, t);
S := {(P, s)}; {S: set of pairs (path, deviation vertex)}
X := {P};
K := {P};
while k <K and X =  do
begin
X := X \ {P};
w := GetDeviationV ertex(S,P);
for v _subP(w, t) _ {t}_ do
begin
G_ := DisableVerticesAndEdges(G, s, v,K,P);
Q := subP(s, v)  SPD(G_, v, t);
X := X  {Q};
S := S {(Q, v)}
end;
P := shortest(X);
K := K {P};
k := k + 1
end
end {procedure}
```

B. Experiment



Step 0: Q1 := _A,B, Z_, |Q1| = 20, Q1 equals P
S := {(Q1,A)}, X := {Q1}, K := {Q1} P1 := Q1
Step 1: X := X \{P1},
1.1: v = A, DE = {1}, DV = ;Q2 = _A,D,C, Z_, |Q2| = 25,
1.2: v = B, DE = {1, 2, 3, 4}, DV = {A}, Q3 = _A,B,C, Z_, |Q3| = 25
S = {(Q1,A), (Q2,A), (Q3,B)}, X = {Q2,Q3}, K = {P1,P2} P2 := Q2
Step 2: X := X \{P2},
2.1: v = A, DE = {1, 3}, DV = ,Q4 = _A,C, Z_, |Q4| = 55,
2.2: v = D, DE = {1, 2, 3, 7}, DV = {A}, Q5 = _A,D, Z_, |Q5| = 30
2.3: v = C, DE = {1, 2, 3, 7, 8, 6}, DV = {A,D}, Q6 = _A,D,C,B, Z_, |Q6| = 40
S = {(Q1,A), (Q2,A), (Q3,B), (Q4,A), (Q5,D), (Q6, C)}, X = {Q3,Q4,Q5,Q6}, K = {P1,P2,P3} P3 := Q3

C. Results of the experiment based on Algorithm:

The length of the shortest path between s and t be equal to p and consider the problem of finding all paths P from s to t with |P| = p in a given graph with non-negative edge weights. As you may notice, Algorithm for finding K-shortest paths can be easily applied for this purpose. This is due to the property that at the end of each iteration, the algorithm adds one path to the current set K = {P1, P2, ..., Pk} of already found k shortest paths. Thus, to find all (and only) optimal paths, we just stop the algorithm when the first path longer than path P1 is found. Based on experiments performed we found that K = {P1, P2, P3} where P1=20, P2=25 and P3=25

VI. RESULTS

This proactive mechanism of calculating the k-shortest path results in saving the valuable time when the load over the network is very high instead the k-shortest path are calculated when the system is idle. The paths get stored in the memories of router and the best path is selected immediately at the time of the failure of the link which reduces the latency in calculating the next shortest path. In case the system not reliable and the selected link again fail then the next shortest path is also available in advance. This increases the reliability of the system by the factor of k. This results in a system that utilizes the system resources by performing some calculations when system is idle and then utilizing that knowledge to reduce the latency for calculating the next available shortest path when the system is at heavy load and failure rate is high. This improves the overall performance of the system and our goal is achieved.

VI. CONCLUSION

The focus of this paper is to investigate the problems experienced by the TCP transport layer protocol and improve performance of TCP protocol, when operating over ad-hoc wireless networks. The key design components of TCP are first highlighted, and for each of the components, the relevance, or lack there-of, with respect to the characteristics of ad-hoc networks was discussed. Then, three major categories of approaches to improve transport layer performance over ad-hoc networks are identified, and giving a solution which is going to fall under the category of the second approach or class and the second approach is TCP-aware Cross-layered Solutions. The trade-offs that stem from the adoption of any one solution is also discussed. The proposed solution which is provided is by showing some sort of proactive behavior and calculating the alternate paths in advance. We calculate k-shortest path in order to perform some proactive measures to reduce the latency for calculating the shortest path at the time of failure and our solution is k times fault tolerant. Higher the value of k more the system will be fault tolerant and most of these calculations are done when the system is idle or when workload is low which increases the system utilization by doing some useful work.

REFERENCES

[1] Transport Layer Protocol in ad-hoc networks, Karthikeyan Sundaresan
[2] K-Shortest Paths and All Optimal Paths, Micha Pióro
[3] Analysis of performance of TCP over ad-hoc network, Gavin Holland and Nitin Vaidya
[4] Performance Comparison of TCP Variants in Mobile ad-hoc networks, Mandakini Tayade, Sanjev Sharma

[5] TCP performance over mobile ad hoc networks, Xiang Chen, Hongqiang Zhai, Jianfeng Wang, and Yuguang Fang
[6] TCP over Wireless Networks: Issues, Challenges and Survey of Solutions M. Patel, N. Tanna, P. Patel, and R. Banerjee
[7] http://www.caida.org/, Co-operative Association for Internet Data Analysis (CAIDA)
[8] Finding the k-Shortest Paths David Eppstein
[9] ATP: A Reliable Transport Protocol for Ad-hoc Networks, K. Sundaresan, V. Anantharaman, H-Y. Hsieh, and R. Sivakumar.
[10] Performance Evaluation of TCP over Mobile Ad-hoc Networks, Foez ahmed, Sateesh Kumar Pradhan, Nayeema Islam, and Sumon Kumar Debnath.
[11] Routing Load of Route Calculation and Route Maintenance in Wireless Proactive Routing Protocols, D. Mahmood, N. Javaid, U. Qasim‡, Z. A. Khan, University of Alberta, Alberta, Canada.
[12] Routing, Flow, and Capacity Design in Communication and Computer Networks, Micha Pióro Warsaw University of Technology, Warsaw, Poland Lund University, Lund, Sweden Deepankar Medhi University of Missouri-Kansas City Kansas city, Missouri, USA.
[13] Impact of Routing and Link layers on TCP Performance in Mobile Ad-hoc Networks, in Proceedings of IEEE WCNC, New Orleans, September 1999. G. Holland and N. H. Vaidya.
[14] TCP Performance in Wireless Multi Hop Networks, in Proceedings of IEEE WMSCA, New Orleans, Feb 1999, M. Gerla, K. Tang, and R. Bagrodia.
[15] An Analysis of Short-term Fairness in Wireless Media Access Protocols (poster), in Proceedings of ACM SIGMETRICS, Measurement and Modeling of Computer Systems, C. E. Koksal and H. Balakrishnan.
[16] The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks, in MANET Working Group. IETF, D. Johnson, D.A. Maltz, and J. Broch.
[17] Ad-hoc On-demand Distance Vector (AODV) Routing, in MANET Working Group. IETF, C. E. Perkins and E. M. Royer.
[18] Limitations of TCP-ELFN for Ad hoc Networks, in Workshop on Mobile and Multimedia Communication, Marina del Rey, CA, Oct. 2000, J. P. Monks, P. Sinha, and V. Bharghavan.
[19] A Comparison of TCP Performance over Three Routing Protocols for Mobile Ad Hoc Networks , in Proceedings of ACM MOBIHOC 2001, Long Beach, CA, Oct 2001. J. Liu and S. Singh, ATCP: TCP for Mobile Ad Hoc, T. D. Dyer and R. Bopanna.
[20] NACK Oriented Reliable Multicast (NORM) Protocol Building Blocks, M. Handley, C. Bormann, B. Adamson, and J. Macker