Source Routing made Practical in Embedded Networks

Arijit Ghosh and Tony Givargis
Center for Embedded Computing Systems
School of Information and Computer Science
University of California, Irvine, USA 92697

***Abstract*—Reducing packet latency is an important requirement in embedded networks. Source routing can be used to reduce processing delay at intermediate nodes and thereby reduce the overall packet latency. However source routing is not scalable which makes it unsuitable for larger networks. The addition of the source route to every packet reduces the system goodput (application level throughput). Further, source routes ignore dynamic network conditions which might lead to routing failures. In this paper, we propose strategies to counter these problems. We propose a topology encoding scheme that reduces the overhead and makes source routing scalable. We propose a lazy correction scheme that makes it take cognizance of dynamic network conditions. Through simulations on reasonably large sized network with realistic models for traffic and failure, we show that source routing is indeed usable in practical scenarios.**

I. Introduction

Wired embedded network systems (EmNets), including sensor networks, distributed control applications, and ubiquitous computing environments, often have strict latency requirements. We define network latency as the *one-way* delay incurred by a packet from the time when it is transmitted by the source to the time when it is received by the destination. A major component of the network latency using dynamic routing protocol is the packet processing at each intermediate node. Traversing the different layers of the network stack as well as the routing table lookup at each hop takes up a non-trivial amount of time. We simulated a 4000 node random graph with 100,000 packets routed between 600,000 randomly chosen pairs of nodes. We injected 500 failed links in a manner that 50% of the chosen routes have *at least* one failed link along them. An incredible 93% of the route lookups turned out to be redundant.

To understand the reason behind this, let us consider the case where a single link fails. Let there be a source node *A* and a destination node *B*. Let the link between nodes *C* and *D* fail. Let P_{old} and P_{new} be the paths between *A* and *B* before and after the link fails. If $CD \notin P_{old}$, then $P_{old} = P_{new}$ and all intermediate lookups will be redundant. Let us now consider the case where *CD* is in P_{old}. Then obviously, $P_{old} \neq P_{new}$. As the link state information propagates from *C* and *D* to the rest of the network, all nodes (at any given time) can essentially be divided into two regions: nodes that have received the update and have the newest routes belong to the fresh region; all others belong to the stale region. If

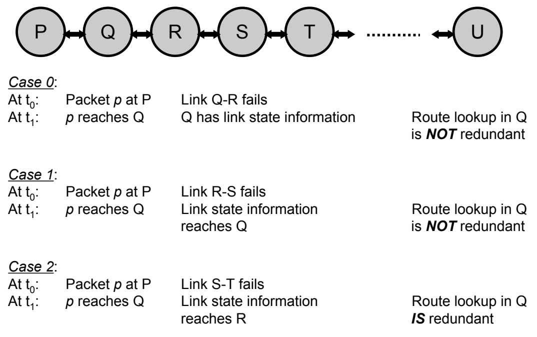

P
Q
R
S
T
U
Case 0:
At t0: Packet p at P
Link Q-R fails
At t1: p reaches Q
Q has link state information
Route lookup in Q is NOT redundant
Case 1:
At t0: Packet p at P
Link R-S fails
At t1: p reaches Q
Link state information reaches Q
Route lookup in Q is NOT redundant
Case 2:
At t0: Packet p at P
Link S-T fails
At t1: p reaches Q
Link state information reaches R
Route lookup in Q IS redundant

Fig. 1. Redundancy: example topology

A is already in the fresh region, then *A* will generate P_{new} making all intermediate lookups redundant. Now consider the case when *A* is in the stale region. *A* will generate P_{old} as the route of the packet. All intermediate nodes in the stale region will produce the same result as *A* and hence will cause redundant lookups. As soon as the packet reaches a node in the fresh region, P_{new} will be generated. Lookup in this case is *not* redundant. But from now on until *B*, again all lookups will be redundant. Thus, unless the path of a packet traverses the boundary between the two regions, *all* intermediate route lookups will be redundant.

Let *A* be *n* hops away from *B* and generate *r* packets/second. Let *t* and *p* be the propagation time (between consecutive nodes) and the packet processing time respectively. Let *P* be the number of packets that will have exactly one non-redundant route lookup. Then, $P = n \times (t + p) \times r$

Imagine a packet traversing a *n*-hop path from *P* to *U* as shown in Figure 1. For a route lookup at each hop to be non-redundant, all of the following three conditions must strictly hold. One, the Mean Time To Failure (MTTF) between links must exceed the sum of the average packet processing time and propagation time between two consecutive hops of the packet. Two, **all** failures should occur along the path of the packet. Three, the currently failed link can be no farther than two hops away from the current hop. Let the link between *S* and *T* fail when the packet is at *P*. Let the processing and propagation time of both the data and link update packets be the same. Then, when the packet reaches *Q*, the link update will only have reached *R*. Thus, the lookup at *Q* will be redundant.

The alternative is to use source routing where all route computation takes place only once, at the source. This could potentially eliminate a great deal of processing delay. However, source routing is usually considered impractical. In this paper, we analyze a typical EmNet and show that source routing *IS* feasible in practical deployments. The average number of hops between any two pairs of nodes in a 10000 node network is at *most* 6 hops. By employing a topology encoding scheme, we show that this imposes a modest memory requirement of 50KB and a reduction of goodput by less than 1%. We propose a lazy correction scheme that reacts to network dynamics. Finally, we propose a hardware short-circuiting scheme that operates at the link layer to take further advantage of source routing.

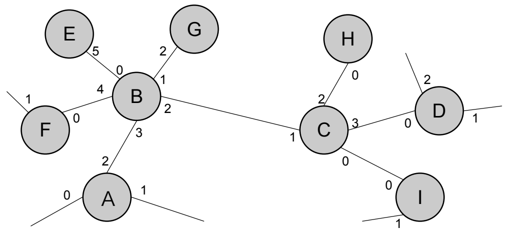
II. Source Routing with Lazy Correction

Source routing is a simple approach where the sender specifies the complete route of the packet along the network. Although simple and attractive, source routing has a few disadvantages. First, source routing is not scalable. Each node has to maintain the path to all nodes in the network which incurs a space complexity of $O(N)$, Further, the entire route is included in the packet header. This reduces the goodput of the system. Second, once a route is specified, it is not changed. By not taking cognizance of changing network conditions, source routing could cause a packet to traverse a longer path. In the worst case, it might even fail to deliver a packet. Third, source routing presents a security hazard since it allows address spoofing. In this paper we address the issues related to scalability and network dynamics and leave security as part of our future work.

A. Scalability

We address the scalability issues by proposing two techniques. The first is a scheme to encode the topology. The second is to build the sensor network according to a particular topology. Neither of the approaches reduce the storage complexity. But our analysis shows that in practical scenarios, the actual number of bytes used will be significantly reduced.

1) Topology Encoding: Let us consider a random topology. Classical random graphs, also known as Erdös-Rényi graphs, are defined by the degree distribution of node *k*, $P(k)$ = exp($-\lambda)\lambda^k/k!$. To uniquely identify *N* nodes in a ER topology, we need $log_2 N$ bits. To reduce the number of bits, we propose an encoding scheme based on logical labels assigned to neighbors. Let *M* be the average number of neighbors of a node in a random graph. In our simple encoding scheme, a node assigns a logical label to each of its neighbor, from 0 to *M*-1. The logical label is independently assigned by each node. When a node joins the network and shares its neighbor information, it includes the logical label corresponding to the neighbor identifier. The source node can now compute the



Node	IP Address
A	A1.A2.A3.A4
B	B1.B2.B3.B4
C	C1.C2.C3.C4
D	D1.D2.D3.D4
E	E1.E2.E3.E4
. . .	. . .

	Route from A to D
Unencoded	A1.A2.A3.A4, B1.B2.B3.B4, C1.C2.C3.C4, D1.D2.D3.D4
Encoded	3, 2, 3

Fig. 2. Path encoding: example graph

route in terms of the labels. Consider the example graph in Figure 2. Let node A be the sender of a packet to node D. Each node assigns a logical label to its neighbor which is indicated next to the edge connecting the node to the respective neighbor. For example, *B* assigns the logical labels 0, 1, 2, 3 and 4 to neighbors *E*, *G*, *C*, *A* and *F* respectively. Under our encoding scheme, the path from A to D is indicated by *2-2-3*. In a random graph where $M \ll N$ [1], this path encoding scheme saves a considerable amount of space (66% for *M* = 1000). The maximum value of *M* in the current Internet is around 2000 [2].

It is well known that the average path length (APL) of ER networks scale in proportion to ln *N* [3]. In [4] it has been analytically shown that the APL is given by

$$APL = \frac{lnN - \gamma}{ln(pN)} + \frac{1}{2}$$

where $\gamma \simeq 0.5772$ is the Euler's constant and $pN = \langle k \rangle$ The APL for ER graphs with $\langle k \rangle$ = 4, 10 and 20 is equal to 6, 4 and 3 respectively for a network with *N* = 10,000. Correspondingly, the diameter *d* of a graph, defined by the maximal distance between any pair of vertices is given by ln *N*/ln(*pN*). From a ER graph of infinite nodes, it has also been shown that 90% of the nodes have an average degree of 5 while less than 0.00001% of the nodes have an average degree of 200 [5]. From the above statistics, we can conclude that a route computed by the source to any node will be bounded by a few hops in a reasonably large network ($\leq$ 6 for a 10000 node network) and that the majority of the logical labels in the path will be small integers.

Our objective is to encode this path with a bit pattern that is quite efficient, is extremely fast to decode and leverages the statistical insights in making the *common case fast* (Amdahl's law). We use the following technique. All labels are classified into two groups Δ_0 and Δ_1. Labels that are less than 32 bits belong to the former group while all others belong to the latter. Accordingly, items in Δ_0 and Δ_1 are encoded by 5 and 8 bits respectively. To distinguish between the labels belonging to the two groups in the path, we prefix a signature of 10 bits in which a 0/1 indicates if the current label belongs to Δ_0/Δ_1.

Although extremely unlikely, it is still possible that a label has a value of more than 255. In this case, we use a hop-by-hop mechanism to negotiate part of the path for which the labels are large. Let a path from node n_1 to n_k be denoted by n_1,n_2, …, n_i, *X*, $n_(i+1)$, … n_j, *Y*, *Z*, $n_(j+1)$, …, n_k, where *X*, *Y* and *Z* are labels grater than 255. In this case, the n_1 computes the partial route to the intermediate node n_i. At n_i, the packet falls back to the default hop-by-hop forwarding mechanism to reach *X*. *X* now computes the partial path to n_j. From here, again the packet gets forwarded one hop at a time up to *Z*. Finally, *Z* computes the rest of the path to n_k. Clearly, falling back to the hop-by-hop forwarding mechanism compromises the speed, However, this is a tradeoff we chose to allow scalability of source routing.

Algorithm 1 Encoder

```
Find the path from source to destination
Specify the path in terms of labels
for each label do
  if label $>$ 255 then
    break;
  end if
  if path[i] $<$ 32 then
    encode with 5 bits
    set corresponding signature bit to 0
  else
    encode with 8 bits
    set corresponding signature bit to 1
  end if
end for
```

Algorithm 2 Decoder

```
if I am destination then
  return
end if
if remaining_path is empty then
  compute remainder path to destination
end if
if path not found then
  drop packet and return
else
  encode(remainder path)
end if
Shift the first signature bit
if bit is 0 then
  label = shift and decode the first 5 bits
else
  label shift and decode the first 8 bits
end if
use label to decide which is the next hop
```

From the above, we can see that the maximum number of bytes to encode the route in a 10000 node network is about 5. It is hard to imagine an EmNet to scale geographically (like the Internet) or in node density (like wireless mote-like sensornets) to have millions of nodes. The space overhead for a 10000 node network is about 50 KB which is modest by current technology standards. Let us consider a TCP/IP packet with a MTU of 1500 bytes. Let the TCP and IP headers each be 20 bytes. For simplicity, let us consider the rest of the packet to contain data. Then an overhead of 5B represents a reduction of less than 1% in the goodput of the system. This shows that source routing is definitely practical in relatively large sized embedded networks.

2) Engineered topology: In mathematics and physics, a small-world network is a type of mathematical graph in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops or steps. Many empirical graphs are well modeled by small-world networks. Social networks, the connectivity of the Internet, and gene networks all exhibit small-world network characteristics. Small-world networks have high representation of cliques, and subgraphs that are a few edges shy of being cliques. The highest-degree nodes are often called "hubs". If a network has a degree-distribution which can be fit with a power law distribution, it is taken as a sign that the network is small-world. These networks are known as scale-free networks. The probability *P(k)* that a node in the network connects with *k* other nodes is proportional to $k^{-\gamma}$. The coefficient γ may vary approximately from 2 to 3 for most real networks [6]. It was proved that an uncorrelated power-law graph having $2 < \gamma < 3$ will also have a network diameter *d* that is proportional to *O*(ln ln *N*) [7]. So from the practical point of view, the diameter of a growing scale-free network might be considered almost constant while the diameter of a 1 million node network is approximately 3 hops. Thus, if a EmNet is built to exhibit the properties of a scale-free network, then it makes using source routing more practical by making the APL almost constant.

B. Network dynamics - Lazy Correction

A second problem of source routing is that it is oblivious to network dynamics. A packet might end up in a "dead end" where the next hop mentioned in the source route is no longer reachable. Routing then fails even though the network is connected. To remedy this situation, we propose a lazy correction scheme. In this strategy, a packet is allowed to move on along the source specified route until it reaches a node n_i which can not reach the next hop $n_{(i+1)}$. In this case, n_i simply discards the rest of the route and recomputes the remainder of the path from itself to the destination. The packet is forwarded along the new route. Since n_i is closer to the destination than the source, it will have a more recent update on the state of the links between itself and the destination as compared to the source. If n_i is unable to find a path, the packet is dropped.

Our approach does not affect the convergence and loop-free characteristics of the underlying routing protocol. Our strategy affects where the rerouting decision will be made and not the route itself. However, it definitely could mean that

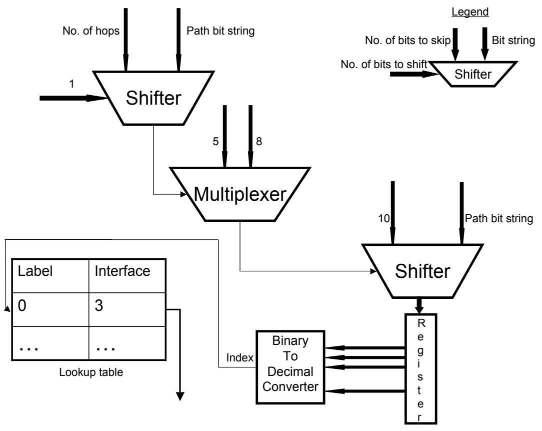


Fig. 3. Short circuiting hardware

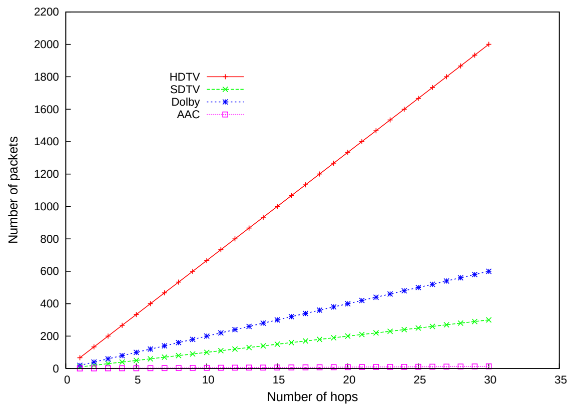


Fig. 4. The number of packets that will incur exactly one non-redundant lookup before all lookups become redundant increases linearly with distance from link failure.

a path travels a longer route than otherwise. We show with experiments that on an average, the path stretch is about 1.13.

III. Short Circuiting

Using source routing provides an opportunity to reduce processing delays at intermediate nodes. However, the benefit can only be had if the packet can be forced to bypass the network layer by being "short circuited" through a lower layer. The idea is to provide an alternate datapath for packets in the node to leverage source routing. Every node maintains a label lookup table which stores the association between a label (and thereby a neighbor) and a physical interface. When a node receives a packet, it checks if the path is empty. If yes, it sends the packet to higher layers. If not, then depending on the signature bit, it shifts out the next 5/8 bits, decodes it to index into the label lookup table and finds the corresponding physical interface. Using a fixed length encoding allows us to build an extremely simple and efficient short circuiting hardware as shown in Figure 3.

IV. Evaluation

We use a two level event-driven simulation. At the session level, the simulator generates flows at a source node. Flows arrive according to a Poisson distribution. We randomly pick flows from one of the following: a single channel MPEG2 encoded HDTV at 20 Mbps [8], a PAL or NTSC-equivalent SDTV at 3 Mbps, a DolbyDigital 'AC-3' audio with a maximum 13.1 channels at 6 Mbps [9] and a standard AAC audio channel at 128 Kbps, At the packet level, it manages the lifetime of a packet. Specifically, it has routing, queueing, failure and delay modules. The routing module simulates a link-state based shortest path algorithm. The queueing strategy used is Worst Case Weighted Fair Queueing (Wf2q) [10]. We choose a failure model described in [11]. To decide *when* the failure happens, we picked a random number from the Weibull distribution with parameters $\alpha = 0.046$ and $\beta = 0.414$. We use the power law with slope = -1.35 to decide *where* the failure occurs according to the following. Link states are flooded only when a link fails. We employ a simple flooding strategy as follows. Upon receiving a link state update (LSU) message, a node updates its database and forwards the information to all its neighbors, except the one from which the update was received. Duplicate messages are discarded. LSU messages are kept in a separate queue and are treated with highest priority by each node. We run our experiments on random graphs generated by Georgia Tech's GT-ITM topology generator. All links are bidirectional. To reduce the number of variables, we use a constant propagation delay of 1ms for all links. All links are assumed to have a capacity of 1 Gbps.

We will evaluate the scalability and effects of network dynamism on source routing. Recall that in source routing, a packet encounters exactly one routing lookup and only when it moves from the stale to the fresh region. To see how many packets, P, encounter a non-redundant lookup, we generated the 4 flows at a source on a linear topology. We induced a single failed link and varied the hop-distance of it from the source. As shown in Figure 4, P increases linearly with the hop-distance as was analyzed in Section 1.

Scalability: We first want to see if the complexity of our encoding scales in proportion to the average path length. We generated different ER networks with up to 10000 nodes. For each topology, we set the average out degree of the nodes to be 4, 10 and 20. We then computed the shortest path between all pairs of nodes and used our encoding technique to compute the number of bits. As can be seen in Figure 5, the increase in the number of bits is definitely $O(\ln N)$. For example, the difference in the number of bits between 2500 and 5000 nodes is 7 (degree = 20). Similarly, the difference between 1000 and 10000 nodes for degree 4 is 10. To evaluate the effectiveness of our encoding, we created 5 topologies with 4000 up to 8000 nodes with unrestricted node degrees. We computed all pair shortest paths and counted the number of bits required for encoding. As can be seen from Fig. 6, 98% of the paths

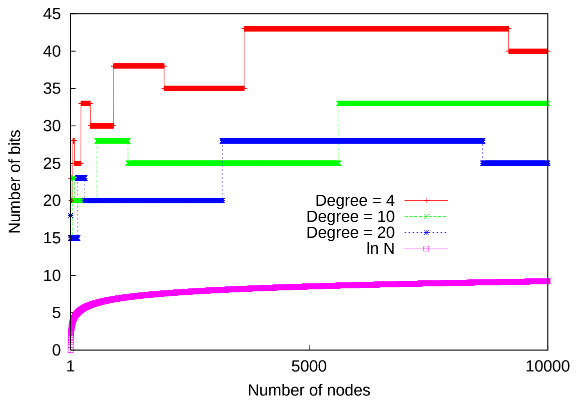


Fig. 5. Scalability of encoding

were encoded with 50 bits or less. Compare this with similar topologies with 32 bit IP-like addresses. The same statistics stood at 250 bits (Fig.7). As can be seen from Fig. 8, even the lowest 5% of the unencoded paths are almost 55% more inefficient (fraction of extra bits used).

Network dynamics: Traditional source routing can fail since the route is never adjusted to reflect the network dynamics. Our approach never fails because we introduce the notion of lazy correction. However, lazy correction does have its drawback. Since it delays its decision to take a better route, there will inevitably be path stretch. Let p_{source} and $p_{dynamic}$ be the paths using source and dynamic routing respectively. We define the path stretch λ_i of a packet i as:

$$\lambda_i = \frac{p_{source}}{p_{dynamic}}$$

The average path stretch, Λ, is the average over all packets. To check this we ran the following two experiments on a 10000 node random topology. We induced up to 500 link failures using our failure model and routed 10000 packets between all nodes that had at least one failed path on their source route. As can be seen from Figure 9, the penalty is not too severe. More than 80% of the paths have a stretch ≤ 1.5. Less than 1% have a stretch ≥ 3.5. On an average though, the path stretch is only 1.13 as shown in Table 1. We next set the number of failed links at a high value of 5000. We generated flows of different duration as indicated by the number of packets. As expected, with increased failures and increased duration of the flows, some packets do show higher path stretch. About 1% of them have a stretch ≥ 3.5 (Figure 10). However, the average path length again is fairly low as shown in Table 2.

In summary, we show in this section that source routing is very scalable. The path can be encoded efficiently with about 50 bits. There is a penalty of path stretch. However, experiments show that it is less than a factor of 1.2. Packet procesiing in the Linux 2.6.9 kernel takes about 15 ms [12]. In contrast, propagation delay for a distance of 1 km is about

TABLE I
AVERAGE PATH STRETCH - 10000 NODES, 10000 PACKETS BETWEEN ALL PAIRS OF NODES

Failures	Average path stretch, Λ
100	1.13381
200	1.13414
300	1.1338
400	1.129
500	1.13

TABLE II
AVERAGE PATH STRETCH (Λ) - 10000 NODES, 5000 FAILED LINKS

Number of packets	Average path stretch
1000	1.13
25000	1.0981
50000	1.09884

3μs. From this, we can conclude that the increase in packet latency for traveling extra hops will be more than offset by the order of magnitude reduction in the processing delay.

V. PREVIOUS WORK

Techniques to improve packet latency have long been a subject of intensive research in the networking community. Many approaches targeting different parts of the networking subsystem have been proposed. There are many service models and mechanisms to reduce packet latency in wired networks. The Integrated Services [13] model is characterized by resource reservation. For real-time applications, before data are transmitted, the applications must first set up paths and reserve resources. In Differentiated Services [14], packets are marked differently to create several packet classes and receive different services. MPLS [15] is a fast forwarding scheme based on labels assigned to packets. Traffic Engineering is the process of arranging how traffic flows through the network. A thorough discussion of all the relevant approaches can be found in [16].

Source routing (SR) is a very old technique that has been used extensively in both wired and wireless networks. In SR, the entire path that the packet travels is included in it. This implies that a node has to maintain a route to every other node. This imposes an $O(N)$ routing table overhead. The typical approach to solving this problem is by introduction of hierarchies [17]. In this paper, we also propose a topological solution but is different from existing hierarchical solutions. The second problem of scalability comes from the increased size of packet headers due to the inclusion of the path. A solution to this was proposed in [18]. In this, the direction of the neighbors are encoded in the source path. We expand on this idea and make it more general so that it can be applied to topologies with arbitrary number of neighbors.

VI. CONCLUSION

In this paper, we have shown that in spite of its perceived shortcomings, source routing is actually usable in practical sized networks. If EmNets are built as scale-free networks and if a topology is encoded in terms of labels, then source routing imposes only a modest overhead on storage requirements and

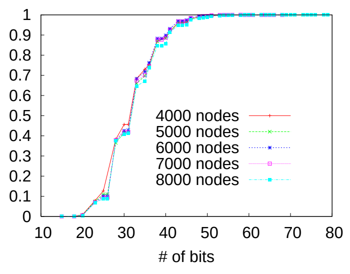


Fig. 6. WITH encoding

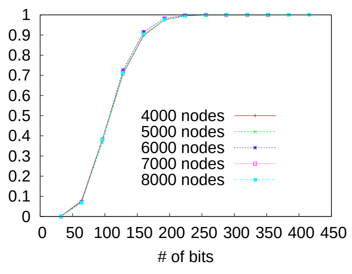


Fig. 7. WITHOUT encoding

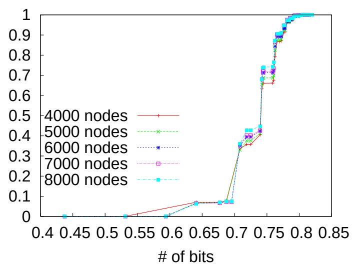


Fig. 8. Ineffciency

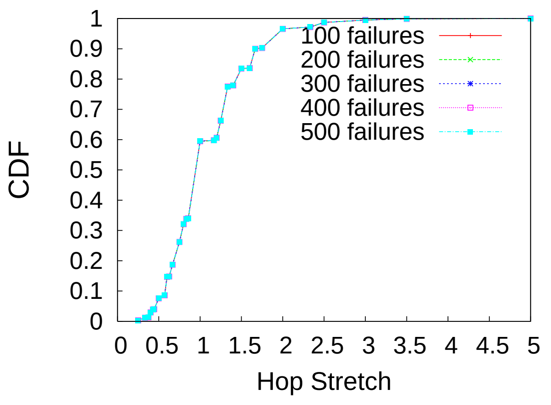


Fig. 9. Effect of number of failures

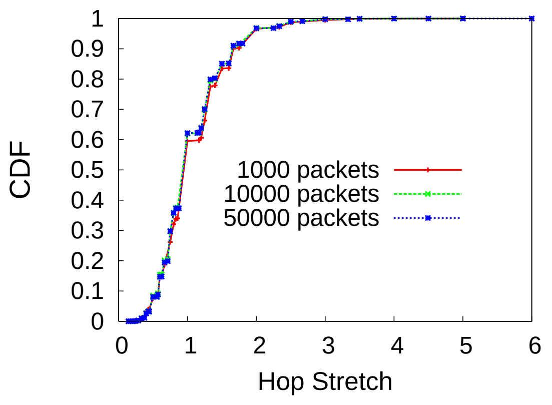


Fig. 10. Effect of number of packets

goodput performance. Source routing allows routing layer functionality to be bypassed at intermediate nodes. We presented a hardware architecture that will allow short-circuiting of a packet through the link layer. As part of our future work, we wish to investigate how source routing can work with different packet scheduling algorithms to reduce latency even further.

Acknowledgment

This work was in part supported by grant #0749508 from the National Science Foundation.

References

[1] D. J. Watts and S. H. Strogatz, "Collective dynamics of small world networks," *Nature*, vol. 393, no. 6684, pp. 440–442, June 1992.
[2] "http://www.caida.org/research/topology/as_core_network/."
[3] M. E. J. Newman, S. H. Strogatz, and D. J. Watts, "Random graphs with arbitrary degree distributions and their applications," *Phys. Rev. E*, vol. 64, no. 026118, 2001.
[4] A. Fronczak, P. Fronczak, and J. A. Holyst, "Average path length in random networks," *Phys. Rev. E*, 2002.
[5] M. Stumpf and C. Wiuf, "Sampling properties of random graphs: The degree distribution," *Phys. Rev. E*, vol. 72, 2005.
[6] H. Seyed-allaei, B. Ginestra, and M. Marsili, "Scale-free networks with an exponent less than two," *Phys. Rev. E*, vol. 73, 2005.
[7] R. Cohen and S. Havlin, "Scale-free networks are ultrasmall," *Phys. Rev. E*, vol. 90, 2003.
[8] "Moving picture experts group, october 2006 http://www.chiariglione.org/mpeg."
[9] "Dolby laborotories inc. www.dolby.com."
[10] J. B. H. and Zhang, "Wf2q : Worst case fair weighted fair queuing," in *INFOCOM*, 1996.
[11] A. M. et. al., "Characterization of failures in an operational ip backbone network," *IEEE Trans. on Networking*, vol. 16, October 2008.
[12] J. Demter and et. al., "Performance analysis of the tcp/ip stack of linux kernel 2.6.9," *Technical Report No. IFI-TB-2005-03, ICS, University of Guttingen, Germany*, April 2005.
[13] R. Braden, D. Clark, and S. Shenker, "Integrated services in the internet architecture: an overview," *Internet RFC 1633*, June 1994.
[14] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *Internet RFC 2475*, December 1998.
[15] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," *Internet draft ¡draft-ietf-mpls-arch-01.txt¿*, March 1998.
[16] X. Xipeng and L. Ni, "Internet qos: a big picture," *IEEE Network*, March 1999.
[17] V. Hadimani and R. Hansdah, "An efficient distributed scheme for source routing protocol in communication networks," *Lecture notes in Computer Science*, vol. 3347, November 2005.
[18] C. Glass and L. Ni, "The turn model for adaptive routing," in *ISCA*, 1992.