Title

Loop-Free Integrated Forwarding and Routing with Gradients

Authors

Garcia-Luna-Aceves, J.J.
Mathewson, James
Ramanathan, Ram
et al.

Publication Date

2018-10-01

Peer reviewed

Loop-Free Integrated Forwarding and Routing with Gradients

J.J. Garcia-Luna-Aceves,*† James Mathewson,† Ram Ramanathan,§ and Bishal Thapa‡
*Palo Alto Research Center, Palo Alto, CA 94304
†Department of Computer Engineering, University of California, Santa Cruz, CA 95064
‡Raytheon BBN Technologies, 10 Moulton Street, Cambridge, MA 02138
§goTenna Inc., 81 Willoughby Street, Brooklyn, NY 11201
Email: { jj, jlmathew }@soe.ucsc.edu, ram@gotenna.com, bishal.thapa@raytheon.com

***Abstract*—Selecting optimum paths subject to multiple constraints is known to be an NP-complete problem for either additive or multiplicative constraints, and very few approaches have been advanced that operate distributively or address more than two constraints. On the other hand, forwarding loops are known to occur in dynamic networks even when routing tables are loop-free at every instant. We propose and analyze IFRoG (Integrated Forwarding and Routing with Gradients), the first approach for loop-free multi-constrained forwarding and routing based on gradient vectors. IFRoG is based on a fully distributed algorithm for the computation of loop-free routes using vectors of gradients that specify path performance for a given additive or multiplicative performance metric (e.g., latency or bandwidth). Data packets are forwarded on a hop-by-hop basis and carry gradient values used to eliminate forwarding loops. We show that IFRoG renders valid loop-free multi-constrained paths to destinations within a finite time and that no data packet can traverse a forwarding loop independently of the state of the forwarding tables maintained by routers. Furthermore, we show that IFRoG has smaller complexity than approaches that require each router to maintain complete network state at each router.**

I. Introduction

As wireless networks have become pervasive in all sectors of society, there has been an increasing need for network-based multimedia applications (e.g., delivering digitized video and audio) to run efficiently over them. For this to happen, solutions are needed for quality-of-service (QoS) routing over dynamic wireless networks, which consists of: (a) finding feasible paths from sources to destinations that satisfy the various application constraints (e.g., bandwidth, reliability, end-to-end delay, jitter); and (b) forwarding the data traffic in a way that network resources are used efficiently. QoS routing differs from conventional best-effort (BE) routing deployed in the Internet and assumed in mobile ad hoc networks (MANET) mainly in that, the path selected for forwarding traffic needs to satisfy multiple constraints simultaneously, and the traffic should be routed in such a way that network resources are used efficiently. This poses a big challenge in dynamic wireless networks because of the inherent mismatch between the stringent QoS requirements of multimedia applications and the physical characteristics of wireless networks, as well as the inherent complexity of routing under multiple constraints. Wang and Crowcroft [13] have proven that the multi-constrained path (MCP) computation is NP-complete when the number of independent constraints is more than one, regardless of whether the constraints are additive (e.g., delay or hop count) or multiplicative (e.g., maximum path bandwidth).

The implementation strategies for QoS routing to date can be classified into two categories: centralized source routing and distributed routing. Centralized MCP algorithms suffer from high computation complexity at source nodes, sluggish response to network changes, and excessive overhead required for the dissemination of topology and resource information throughout the network, which significantly limits their scalability. Distributed MCP algorithms compute feasible paths locally at each node, and forward packets based only on their destination addresses on a hop-by-hop basis.

Many of the existing distributed QoS routing approaches require the availability of timely global network state at each node, with some approaches even assuming that the distribution of routing constraints is known [7]. Furthermore, a large number of schemes address QoS routing subject to only a single constraint (e.g., bandwidth [14], [8]), or compute only the shortest paths with regard to the specified optimization metric and need not satisfy multiple constraints simultaneously [7], [14], [12], [11], [6].

Although much work has been done on QoS routing in the Internet, they cannot be simply applied to mobile ad hoc networks (MANETs) largely due to the dynamic and resource-constrained nature of MANETs. In fact, most proposed routing protocols supporting QoS provisioning for ad hoc networks are derived from existing ad hoc routing protocols (e.g., QOLSR [1], (OLSR) [3], QAODV [9], etc.). Both QOLSR and QAODV mainly focus on bandwidth-delay constrained routing problem, and do not address the general $k\text{-}constrained$ path selection problem.

Sobrinho adopts an algebraic approach and investigates the path optimization problem in the context of Hop-by-Hop QoS routing [12]. Although the results obtained by Sobrinho establish a generalized framework for QoS-oriented path optimization, they cannot be applied to constrained path optimization, because paths are optimized only with respect to the given path weight function, rather than being computed

This material is based upon work supported by the Defense Advanced Research Project Agency (DARPA). The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited by DARPA on April 23, 2018 under DISTAR Case 29311).

to satisfy multiple constraints.

Hence, a solution is needed for distributed multi-constrained routing for wireless networks such that it enables the use of hop-by-hop datagram forwarding rather than source routing, does not require global network state to be made available at each node, and finds multi-constrained paths while optimizing the overall routing performance according to the given optimization metric(s). In this paper, we present Integrated Forwarding and Routing with Gradients (IFRoG) that consists of (a) maintaining the best set of paths to each known destination, where best means that any QoS requirement that is satisfiable by an existing path between a given source and destination is already satisfiable by a path in the set; and (b) forwarding datagrams in a way that the QoS requirements stated in the datagrams are satisfied and no loops are traversed.

Sections II and III describe Integrated Forwarding and Routing with Gradients (IFRoG). Section IV discusses why iFRoG provides loop-free packet forwarding in iFRoG and Section V addresses its complexity. Section VI presents preliminary simulation results addressing the performance of iFRoG.

II. Preliminaries

QoS-related routing metrics, as well as the constraints associated with them, can be categorized into *minimal* (or concave), *maximal* (or convex), *multiplicative* and *additive* metrics. The path measurement of a minimal metric (e.g., bandwidth) is determined by the minimal value of this metric of all links in the path. The path measurement of a maximal metric (e.g., probability of interceptability) would be inverse of a minimal metric. The path measurement of an additive (e.g., delay) or multiplicative metric (e.g., loss rate) equals the sum or product of its values of all links along the path.

Multiplicative metrics, such as loss rate, in which the end-to-end path loss is equal to the product of the loss rates of all intermediate links, can be translated into additive metrics, or vice versa, by taking logarithmic or exponential function, respectively. Therefore, we only consider minimal and additive QoS metrics and their associated constraints in this work, unless specified otherwise.

The *logical distance (LD)* of path p is given by a path function (or an optimization function) f^p based on the weights of its consisting links.

It is important to point out that logical distances are **not** necessarily simple real numbers. A logical distance is a real number only if the optimization function f^p can be given by a close-form expression. This is the case for the aggregated metric used in the IGRP [4]: $f^p = L + \frac{k}{C}$, where k is a positive constant, L and C are the path length and capacity, respectively.

In general, a logical distance can be viewed as a tuple consisting of multiple routing metrics. For example, for widest shortest-path (WSP) routing, a path with the shortest distance (e.g., hops) is preferred, and if multiple such shortest paths exist, the one with the maximal bandwidth is preferred. The f^p for WSP is then defined by $(\sum D_{u,v}, min(B_{u,v}))$, in which $D_{u,v}$ and $B_{u,v}$ are the distance and bandwidth of each comprising link $l_{u,v}$ along the path. As a result, to compare the precedence between two paths, only bandwidth or distance is not enough. Tuple $< D, B >$ has to be used as the logical distance for each path (D and B are distance and bandwidth, respectively), and path p with logical distance $< D1, B1 >$ is better than path q with $< D2, B2 >$, only if $D1 < D2 \vee (D1 = D2 \wedge B1 > B2)$.

Similarly, in least-cost shortest-path routing (LCSP), tuple $< D, C >$ is used as the logical distance for each path, in which D is the distance and C is the cost. Function f^p is defined by $(\sum D_{u,v}, \sum C_{u,v})$ over all comprising link $l_{u,v}$ of a path. Path p with $< D1, C1 >$ is preferred over path q with $< D2, C2 >$, only if $D1 < D2 \vee (D1 = D2 \wedge C1 < C2)$. By extrapolating the real-number metrics used in conventional best-effort routing to logical distances, it is handy to compute optimal paths in the context of QoS routing, provided that a total order properly exists amongst the logical distances defined by the optimization function f^p being used.

III. IFRoG

Assumption: IFRoG assumes that every node detects, within finite time, the existence of a new neighbor or the loss of connectivity with an existing neighbor node; all packets transmitted over an operational link are received without error within a finite time; all routing messages, changes of link weight or status (up/down), are processed one at a time and in the order in which they are received or occur.

In IFRoG, each physical radio or interface (wired or wireless) has a tuple of MCP constraints. There may be multiple interfaces on each node, and each tuple is associated with a unique interface address. When each node initially broadcasts a GEM (Gradient Establishment Message) packet, via UDP, it only contains information identifying the name of the source node, the IP address associated with the sending interface, a sequence number of the GEM packet, and a tuple of k-constraints. When the GEM gradient packet is received, the values are matched to the receiving interface, using the interface-sending address pair to identify gradient path. It is important to note, when a node sends out a GEM packet, the gradient values are set by the receiving interface (not the sending interface). This is an important distinction noted later, in the results. Algorithm 1 shows the steps to process the GEM packets. GEM packets are flooded, but only propagated if they are newer (via sequence number) or they dominate the MCP in the current entry. GEM packets mark threshold values from the neighbor as a forwarding route (if those constraints are met). The forwarding table is kept fresh by purging all entries more than 2 sequence numbers old.

Algorithm 2 shows the steps to forward a packet, based upon information from the GEM packets. Each route entry with a destination address is placed in a list. Then, each constraint is applied to the list, until a destination is found. If a destination is not found, due to the constraints, the packet is dropped (even if a non constrained path exists). The table makes sure all paths meet the minimum thresholds required, except in the case of the most dominant value (threshold=0). In the event

Algorithm 1 Process GEM Packets

```
Constraint = GEM_{Gradient}(f^p) (Interface_{gradient} Property)
GEM_{Gradient} ← Constraint
Key←tuple(Node_{name}, Neighbor_{IpAddress})
NameIpAssociationTable←(Node_{name}, Node_{IpAddress})
if Key does not exist in ROUTING_TABLE then
    ROUTING_TABLE_{IncomingInterface} ←(Key, Tuple(Constraint, TIME_{NOW}(), GEM_{SeqNum})
    RESEND_GEM(GEM)
else
    for all Entry in ROUTING_TABLE matching Key do
        if Tuple(SeqNum) < (GEM_{SeqNum} − 2) then
            REMOVE ROUTING_TABLE_{ENTRY}
            DROP_PACKET()
        else
            Gradient_{Entry} ←ROUTING_TABLE(GEM_{SeqNum}
            if GEM_{Constraint}.IsDominate(Gradient_{Entry}) then
                Gradient_{Entry} ←GEM_{Constraint}(f^p)Gradient_{Entry}
                ROUTING_TABLE ← Gradient_{Entry}
                RESEND_GEM(GEM)
            else
                DROP_PACKET()
            end if
        end if
    end for
end if
```

there is more than one candidate which meets the constraint requirements, the forwarding path is chosen at random from the candidates.

Algorithm 2 Forwarding Packets via IFRoG

```
Node_{destination} ←NameIpAssociationTable(GEM_{IpSrc})
Constraint←GEM_{Gradient}
Threshold←GEM_{Tos}
if Node_{destination} not found then
    DROP_PACKET()
else
    NeighborList←ROUTING_TABLE(Node_{destination})
    for all constraint_{type} in Constraint do
        REDUCE_LIST(NeighborList,constraint_{type}, threshold)
        if NeighborList.size=0 then
            DROP_PACKET
        else if NeighborList.size == 1 then
            return
        end if
    end for
    return Forward Packet to Random Selected Dest in NeighborList
end if
```

Algorithm 3 reduces a potential list of forwarding candidates, by using the appropriate thresholds and comparison functions.

Technically, there is no hard limit on how many constraints can be transmitted in a packet. The limitations could come from the upward facing Application Interface. For example, if the Application Interface was IP (e.g., IPv4), the 8 bits TOS field limitation would restrict how much constraints can be served simultaneously.

In the first implementation of IFRoG, we define the QoS requirement with the 3-constraint gradient: Bandwidth, interceptability, and hops. Bandwidth is represented as a 32-bit value (bps), interceptability is an arbitrary 8-bit value, where a large values indicates it is very easy to intercept the signal, and a low value indicates incremental difficulty in intercepting the signal (e.g. directional waveforms). Hops is the traditional metric used to indicate distance between nodes.

The first GEM packet is broadcast over a radio or interface,

Algorithm 3 Threshold Constraint Reduction

```
DEF FN() REDUCE_LIST(NeighborList,constraint_{type},threshold)
gradientMaxValue=MinimalDominatedValue
for all element in NeighborList do
    if element.dominates(threshold,(f^p)) then
        gradientMaxValue = element
    else
        Erase(element from NeighborList)
    end if
end for
if Neighborlist.size < 2 then
    return
end if
if threshold=0 then
    for all element in NeighborList do
        if element ≠ gradientMaxValue then
            Erase(element from NeighborList)
        end if
    end for
end if
return Forward Packet to Random Selected Dest in NeighborList
```

it does not contain any constraint gradient information. When a packet is first received, the receiving nodes applies its gradient tuple information to the packet, and rebroadcast over its radio. Each node keeps a separate table, with neighbor-gradient stored in tables.

As each GEM is received, it is added to the table, if it is from a new neighbor (as seen on that interface).

A GEM is dropped if it is inferior in all gradient tuples compared to the existing record in the table. As an example, a node may receive a GEM packet with a gradient field hop of 4, and a bandwidth of 8Mbps. This same node may receive another GEM packet, from the same source and intermediate neighbor, with a hop of 8, and a bandwidth of 50Mbps. The hop count is inferior, and if it was the sole means of comparison, it would be dropped. But the bandwidth field is dominated (50 Mbps >8 Mbps), so the table going to the source node, through the neighbor, would be marked as hop:4, bw:50Mbps. The node will route all packets, needing those requirements, to the neighbor node, knowing the neighbor will have 1 or many potential paths to route the packet. No computation is needed, only newer GEM packets are saved or current routing information is updated.

IV. Loop-Free Forwarding Using Gradients

To avoid loops, all gradient fields in GEM are saved (if newer or dominated). If an update occurs, where a dominant value in the MCP exists for the same source-neighbor-interface tuple, it will replace the the entry if it is totally dominated (all fields are dominated). If only a single field is dominated, the old entry is replaced with the updated information. This is done to avoid routing loops, which follow the trail of bread crumbs from the GEM packets to the destination. Since each entry is saved, when a particular entry is selected, the TTL field in the IPv4 packet, for example, is modified with the expected hop count, as opposed to a much larger default. The modification only occurs in the source node.

The GEM packets have a built in TTL, as well as processing to avoid flooding inferior source packets. Nodes only flood GEM packets which are updates (by sequence number), or

have, at least, a single dominating value in its gradient payload.

Two approaches are currently used to cope with the occurrence of forwarding loops in datagram forwarding. In the forwarding plane, the TTL field of a datagram is used to discard a datagram after it circulates a forwarding loop too many times. In the control plane, routing protocols (e.g., OSPF and EIGRP) are used to reduce or eliminate the existence of routing loops. However, even if no routing loops ever occur in the control plane, a datagram may still circulate along a forwarding loop while routing tables are inconsistent among routers. In IFRoG, Forwarding Information Base is used to ensure that forwarding decisions in the data plane are consistent with the routing information maintained by the routing protocol operating in the control plane.

The FIB entry stored at router i for address d states the minimum-hop distance ${H_d}^i$ to the address in addition to the next hop n to it. Router i uses the following rule to forward a datagram using its FIB within a network.

TTL-based FIB Rule (TFR): Router i accepts to forward a datagram from router k towards the best-match for d if TTL $>$ ${H_d}^i$. A router simply drops a datagram intended for a destination address of global scope with a TTL value that does not satisfy TFR.

Theorem 1 proves that TFR eliminates forwarding loops.

Theorem 1: No datagram can traverse a forwarding loop in a network in which TFR is used to forward datagrams.

Proof: We denote by $P^k[s, d, T^k](p)$ a datagram sent by router k with a header that contains a source address s, a destination address d, a TTL value (T^k), plus payload data p. Router i uses the following rule to forward such a datagram using its FIB within a network. Consider a network in which TFR is used and assume for the sake of contradiction that routers in a forwarding loop L of h hops $\{v_1, v_2, ..., v_h, v_1\}$ forward a datagram for destination d along L with no router in L detecting that the datagram has traversed loop L.

Given that L exists by assumption, router $v_k \in L$ must forward $P^{v_k}[s, d, T^{v_k}](p)$ to router $v_{k+1} \in L$ for $1 \leq k \leq h-1$, and router $v_h \in L$ must forward $P^{v_h}[s, d, T^{v_h}](p)$ to router $v_1 \in L$.

According to TFR, if router v_k ($1 < k \leq h$) forwards $P^{v_k}[s, d, T^{v_k}](p)$ to router v_{k+1} as a result of receiving $P^{v_{k-1}}[s, d, T^{v_{k-1}}](p)$ from router v_{k-1}, then it must be true that $T^{v_{k-1}} > H_d^{v_k}$. Similarly, if router v_1 forwards $P^{v_1}[s, d, T^{v_1}](p)$ to router v_2 as a result of receiving $P^{v_h}[s, d, T^{v_h}](p)$ from router v_h, then it must be true that $T^{v_h} > H_d^{v_1}$. However, these results constitute a contradiction, because they imply that $H_d^{v_k} > H_d^{v_k}$ for $1 \leq k \leq h$. Therefore, the theorem is true. ∎

TFR consists of imposing an ordering constraint on the traditional datagram forwarding algorithm based on FIB entries, and making the TTL value of the datagram equal to the distance stored in the FIB for the intended destination, rather than simply decrementing its value. The result of Theorem 1 is independent of whether the network is static or dynamic, or the type of routing protocol used to compute distances and next hops. The ordering constraint of TFR is essentially the same loop-free condition first introduced in DUAL [5]. The difference between the way in which the ordering constraint is used in TFR and in DUAL is that TFR establishes distance-based ordering in the data plane to forward datagrams based on existing FIB entries, while DUAL establishes distance-based ordering in the control plane to build FIB entries that are then used to determine how to forward datagrams.

V. Performance Implications

Under IFRoG, if up to x non-dominated paths are maintained for each destination at any given router i, the space complexity is $O(x|N^i|N + xN) = O(x|N^i|N)$, where $|N^i|$ is the number of neighbors of node i, because the main routing table and each neighbor table have $O(N)$ entries, and each entry can keep up to x routes for each destination.

In practice, the number of non-dominated paths between source and destination can be exponential [15], and Yuan [15] has shown that $O(N^2 \log(N))$ non-dominated paths need to be maintained to have high probability of finding feasible paths. However, if the data flows induced by the applications are not such that their individual bandwidth and delay demands are close to the total available bandwidth and average end-to-end delay over an average path, a satisfactory success ratio can be attained in IFRoG with a relatively small value of x, i.e., of order $O(1)$ with respect to N.

The computation complexity of the time needed for a router to process gradient vectors regarding a particular destination is $O(x|N^i|)$. When only the single shortest path is maintained, then the space and computation complexity reduce to $O(|N^i|N)$ and $O(|N^i|)$ respectively, which are the same as that of the distributed Bellman-Ford algorithm (DBF) [2].

The time complexity of IFRoG is the time it takes to converge after a single change in the network, and the communication complexity is the amount of messages required to propagate this change before all routers can integrate it and update their routing tables accordingly. Given that IFRoG is free of routing-table loops, any routing information (GEM) propagates as fast as the shortest physical path that dominates other paths with respect to the gradient metric between its origin and the recipient. This means that the time complexity of IFRoG is $O(l)$, where l is the length of the longest path that a GEM containing a tuple that corresponds to a dominating path along which a given gradient metric must traverse. The length of l depends on the topology of the network and the performance characteristics of its links. In the worst case, its length could be $N-1$; however, it is much closer to the network diameter in practice.

An update message (GEM) in IFRoG contains all the tuples needed to update the gradient vectors for a given destination. Accordingly, the number of messages required for all routers to have a correct logical distances to a given destination is $O(N \times E)$, where E is the number of links in the network and N is the number of destinations.

VI. Preliminary Simulation Results

To simply verify the IFRoG design, a number of simulation scenarios were designed, to enable easy visual verification of

correct behavior. NS-3[16] was chosen as the simulator, using standard sockets and IPv4 as the transport layer (with the TOS field enabled). A six node scenario 1 using parameters shown in Table I was run to demonstrate the key capabilities of IFRoG. Table I shows each named interface (e.g. A1, A2, B1, B2, ...) and the corresponding bandwidth and interceptability of each interface.

The first 3 tests are routing tests that use each primary gradient value (bandwdith, interceptability, and hop distance), using a threshold value of zero (or use the largest value, and if a match, continue the partial order determination) to define the metric. The first 3 tests have similar parameters, with traffic flowing from node 5 to node 0, based upon a specified routing MCP (bandwidth, hops, and interceptability, respectively).

TABLE I
SIMULATION PARAMETERS

Node Connect	Node_1 BW	Node_2 BW	Node_1 Interc	Node_2 Interc
A	210	80	80	10
B	220	250	40	40
C	250	250	250	250
D	250	250	250	250
E	250	250	150	140
F	80	80	10	10
G	210	210	210	110
H	90	150	140	140

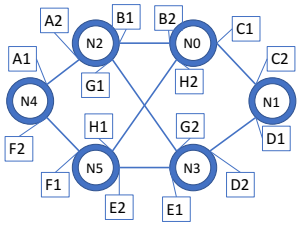

Fig. 1. Small scale network

Figure 2 illustrates routing by the highest-bandwidth parameter, going in a counter clockwise direction, from node $5 \rightarrow 3 \rightarrow 1 \rightarrow 0$. To validate routing by minimum-hop count, Figure 3 shows the expected routing, going up from node $5 \rightarrow 0$. The final parameter, Interceptability, follows the path of least interceptability, taking a clockwise direction, going from node $5 \rightarrow 4 \rightarrow 2 \rightarrow 0$ as shown in Figure 4.

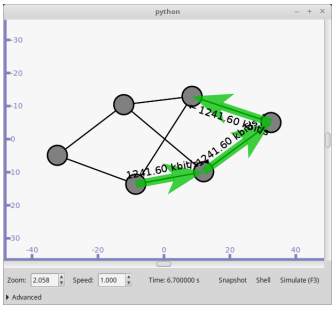

Fig. 2. Routing by highest bandwidth

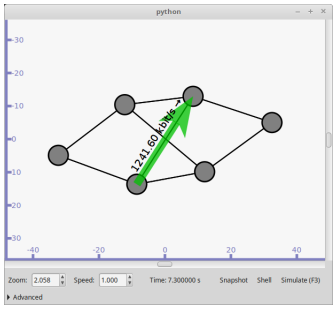

Fig. 3. Routing by lowest hop distance

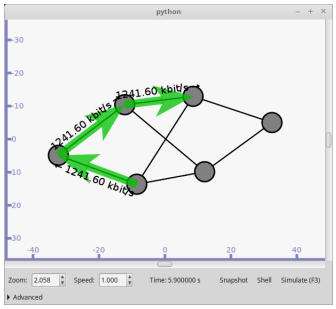

Fig. 4. Routing by minimum interceptability

Routing with multiple constraints, does not require the constraints to be real; they can be virtual. In Figure 5, we demonstrate maximum-bandwidth usage by creating a single virtual ethernet channel. Node 5 transmits three different types of packets, each carrying a different TOS value. One TOS value directs the path to take a clockwise direction, another goes counterclockwise, and the last one takes a direct route up (using the bandwidth, interceptability, and hop count parameters). Actual values, as reported to the GEM packet, may, or may not, reflect actual hardware implementations, but they mimic simple traditional TOS service, each taking a different path. As an example, the highest priority packets could take the 1-hop path, while a high value customer takes a high bandwidth route, with general customers assigned to a lower bandwidth route. By establishing different parameters as a more traditional QoS routing, it can remove some of the headache of having multiple constraints which may not directly apply to a particular set of network traffic.

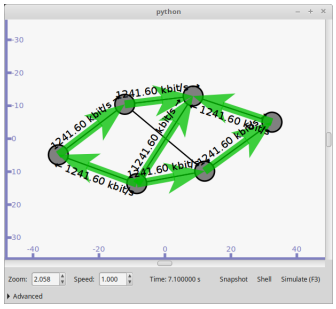

Fig. 5. Load balancing

Figure 6 is an example of using minimal thresholds. Using a gradient TOS requirement of Bandwidth (threshold index of 3, or 200Mbps+), Hops, and finally Interceptability (threshold index of 2, for 150Mbps-). While $N5 \rightarrow B2$ has 2 paths meeting the bandwidth requirement (both considered equal), the constraint goes to a second order of hop distance, which both paths are equal in (3 hops). Finally, the constraint goes to the third order, which only the path from Node $5 \rightarrow 3 \rightarrow 2 \rightarrow 0$ meets the requirement. The threshold values are set per node. Note that this opens up future work, by allowing traffic, which may not be allowed by the network constraint thresholds.

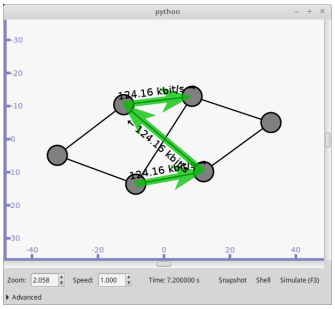

Fig. 6. Routing by Bw 5Mbps+, Interc 100-, Hop Count

As a reminder, the path is set from the node destination (Node 0, in this case), propagating GEM messages to be received by Node 5. In this example, there are 4 paths for the 3 interfaces. Node 0 sends a GEM, and the interface C2 (node 1) applies the dominant gradient rules. It will then broadcast to node 3 (D2 interface), and apply the dominate gradient rules. Finally, it will arrive at node 5 (interface E2), and the path from node 5, to node 0, will use the interface rules from E2, D2, and C2; along one path). Likewise, node 5 will broadcast GEMs that will take path (E1, D1, C1), in addition to other paths. Figure 7 shows this asynchronous nature, by having node 5 transmit to node 0, and node 0 will transmit to node 5, using the highest bandwidth value. Instead of being forced for a packet to go from node $5 \rightarrow 3 \rightarrow 1 \rightarrow 0$, it still follows the highest bandwidth value from node 0 to 5, via nodes $0 \rightarrow 2 \rightarrow 4 \rightarrow 5$.

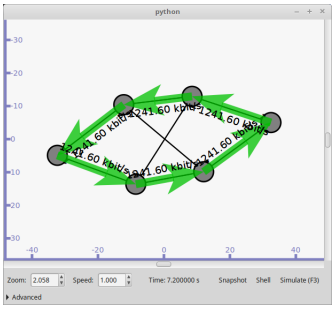

Fig. 7. Asyncronous Routing

VII. Conclusions

We introduced a simple approach for integrated forwarding and routing with multiple QoS constraints that ensures that datagrams are sent over paths that satisfy the QoS desirements specified by applications, and eliminates forwarding loops even when routing tables have inconsistent forwarding state. We implemented the QoS route computation of IFRoG in NS-3 using IPv4 datagrams with the TOS filed enabled as the network layer, and verified that the paths computed with IFRoG satisfy the desired QoS requirements stated by applications. IFRoG can successfully support k-constraint routing with threshold limitations and multiple radios or interfaces per node. We note that the preliminary results presented in the paper, by no means, neither are nor meant to be the product of extensive performance evaluation tests. They are results of the tests run to simply verify the design and demonstrate key capabilities without stress testing the algorithm in a larger and/or a more dynamic network setting. Our future work focuses on the performance of IFRoG in dynamic wireless networks, the implementation of the loop-free forwarding component of IFRoG, and a more flexible specification of the QoS requirements in datagrams.

References

[1] H. Badis and K. Agha, "Quality of Service for Ad-hoc Optimized Link State Routing Protocol (QOLSR)," IETF Internet Draft, March 2006.

[2] D. Bertsekas and R. Gallager, *Data Networks - 2nd Edition*, Prentice-Hall, 1992.

[3] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," RFC 3626 (Experimental), October 2003.

[4] J. Doyle, *Routing TCP/IP*, Cisco Press, 1998.

[5] J.J. Garcia-Luna-Aceves, "A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States," *Proc. ACM SIGCOMM '89*, Aug. 1989.

[6] Z. Li and J. J. Garcia-Luna-Aceves, "A Distributed Approach For Multi-Constrained Path Selection And Routing Optimization," *Proceedings of QSHINE'06, Waterloo, Ontario, Canada*, August, 2006.

[7] P. V. Mieghem, H. D. Neve, and F. Kuipers, "Hop-by-Hop Quality of Service Routing," *Comput. Networks*, 37(3-4):407–423, 2001.

[8] S. Nelakuditi, Z. Zhang, R. P. Tsang, and D. H. C. Du, "Adaptive Proportional Routing: A Localized QoS Routing Approach," *IEEE/ACM Trans. Netw.*, 10(6):790–804, 2002.

[9] C. Perkins, E. Royer, and S. Das, "Quality of Service in Ad-hoc On-demand Distance Vector Routing," IETF Internet Draft, July 2000.

[10] C. Perkins, E. Royer, and S. Das, "Ad Hoc On Demand Distance Vector (AODV) Routing," RFC 3561, July 2003.

[11] B. Smith and J. J. Garcia-Luna-Aceves, "Efficient Policy-Based Routing without Virtual Circuits," *Proceedings of QSHINE'04, Dallas, Texas*, October, 2004.

[12] J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," *IEEE/ACM Trans. Netw.*, 10(4):541–550, 2002.

[13] Z. Wang and J. Crowcroft, "Quality-of-Sservice Routing for Supporting Multimedia Applications," *IEEE Journal of Selected Areas in Communications*, 14(7):1228–1234, 1996.

[14] J. Wang and K. Nahrstedt, "Hop-by-Hop Routing Algorithms for Premium-class Traffic in Diffserv Networks," *Proceedings of IEEE INFOCOM*, 2002.

[15] X. Yuan, "Heuristic Algorithms for Multiconstrained Quality-of-Service Routing," *IEEE/ACM Trans. Netw.*, 10(2):244–256, 2002.

[16] https://www.nsnam.org/.