2009 Eighth IEEE International Symposium on Network Computing and Applications

Investigating the use of Topology Adaptation for Robust Multi-Path Transport: A Preliminary Study

Marco Arguedas, Carlos Perez, Marco Carvalho, Adrian Granados
Institute for Human and Machine Cognition
40 S. Alcaniz St., Pensacola, FL 32502
Email: {marguedas,cperez,mcarvalho,agranados}@ihmc.us

Kelli Hoback, Wayne Kraus
Rockwell Collins
400 Collins Rd NE, Cedar Rapids, IA
Email: {kahobak,wakraus}@rockwellcollins.com

***Abstract*—In this paper we introduce a topology control algorithm for increasing transport robustness and efficiency by creating two simultaneous communication paths between the source and destination nodes of a data flow. We make use of a cross-layer substrate that allows us to detect the flows of data in the network while also providing finer control of the routing. Because information of flow traffic in the network is available, the algorithm attempts to achieve intra-flow interference reduction by exploiting node mobility.**

I. Introduction

Several routing protocols that try to maximize network connectivity in mobile ad-hoc networks (MANETs) have been proposed [1], [2], [3] in recent years. However, some MANETs might have other concerns such as load balancing, fault-tolerance and/or improvement in bandwidth on top of node connectivity, and multi-path routing is one of the preferred mechanisms for addressing these issues. In this paper we present *dpath*, a distributed topology control algorithm designed to create, on demand, two simultaneous data communication paths between the source and destination nodes of a data flow. The algorithm creates the interference free data paths by moving nodes for the duration of the data flow. The algorithm adjusts the position of the nodes to create two data paths that are both link and edge disjoint, thus interference free under certain simplifying assumptions[1]. Given enough resources, the only two points of contention between the data paths are the source and destination nodes.

Once the disjointed path has been created, the sender node may choose to utilize both paths to improve robustness by duplicating the transmission of each data packet over the paths. Alternatively, the sender node may chose to improve throughput instead by interleaving the transmission over each link. Figure 1 shows an instance of a network with disjointed paths between nodes '0' and '11'.

[1]This is strictly true for a unit disc topology. In most practical cases, however, the transmission interference range is significantly higher than the connectivity range (defined by the RSSI threshold) which would technically require the algorithm to be defined over two topologies, one representing the network connectivity and one for interference (a contention graph). Nevertheless, as currently proposed, the algorithm relies on the simplifying assumption that network propagation can be approximated to a unit-disc, but we expect that it will well tolerate small violations of the assumptions.

II. Related Work

Common choices of multi-path routing protocols include SMR (Split Multipath Routing) [4] and AODVM (Ad hoc On-Demand Distance Vector Multipath) [5]. Both of these protocols are on-demand (*reactive*) routing protocols. These modified versions of DSR[1] and AODV[2], respectively, have been adapted to handle multiple node-disjoint paths. In these algorithms, the selection of the disjointed path is realized at the destination node. Such decision is based on global information gathered from and provided by route-request messages.

These algorithms, however, are bound by a common set of assumptions. First, the algorithms consider that the topology is fixed, most likely because neither the mobility of the nodes or the transmission power of the devices can be controlled. Second, the mechanisms of route creation and selection do not take into consideration historical information of data traffic.

In [6], the authors attempt to improve network throughput by making use of a cross-layer approach. However, the presented approach tries to solve the multipath routing problem using genetic algorithms, a solution that might be too complex to be performed in real-time.

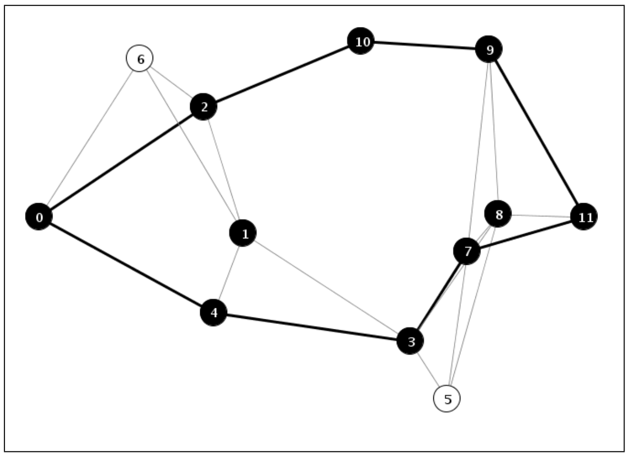

Fig. 1. Dual disjoint path in place, allowing sender to choose between end-to-end robustness and throughput.

III. The Dual-Path Topology Control Algorithm

The *dpath* algorithm is a reactive algorithm, triggered by the presence of a data flow in the network. The algorithm is incremental in nature, in the sense that changes in topology are progressively reinforced by each packet in the data flows. The approach allows for a solution that evolves with other uncorrelated changes in the network topology or data flow. There are no requirements for the duration of the data-flow (either in number of packets or time) for triggering the protocol.

In its current implementation, the topology adaptation rate is fixed (i.e. the rate in which the topology adapts to an existing data flow is fixed and independent of the rate of the flow). However, a variation of the protocol can be easily conceived to increase the adaptation rate as a function of the data rate, forcing the algorithm to react accordingly to different types of flows.

A. Data Flows

In the context of this work, a data flow is identified by a pair (n_s, n_d), where n_s is the source node, and n_d is the destination node of *unicast* data packets. At any given time, multiple data flows may co-exist in the network and, under the assumption that all transmissions occur over the same frequency (no spectrum allocation), to minimize interference, a given node should not participate in more than one of the paths of any given data flow. The goal of the algorithm is to separate each data flow into two disjoint data paths between source and destination.

For instance, in Figure 1 there is a single data flow between nodes '0' and '11' going through two non-interfering data paths (represented with thicker lines). The current version of the algorithm defines a flow based solely on source and destination addresses, disregarding traffic type and port numbers. Multiple independent data flows between a given source and destination (for instance between different applications) will be reduced to the same *flow*, as they share the same source and destination IP addresses. This simplification is acceptable because a dual communication path between each node is valid regardless of the associated port numbers involved.

The choice of constraining the protocol to unicast traffic is to ensure that no adaptation takes place over broadcast messages, which are commonly used for network management and routing protocols.

B. Node State

Every node in the network maintains a list of the data flows in which it participates (that is, the node is either the source, destination, or a forwarding node of the data packets of a flow). In addition to their identifiers (source and destination node addresses), the flows are annotated with the immediate last-hop node from which it was received (n_p), the set of nodes used as immediate next-hop for that flow (N_f), and a binary flag (s) indicating if the data flow has been split (i.e. forwarded to more than one next-hop neighbor) by any of its upstream nodes, including the source node.

$\mathcal{F}_n$ constitutes the set of metadata pieces associated with all the flows going through a node. Each node creates and maintains this metadata based on information relayed with the data packets. $\mathcal{F}_n$ is defined as a set of tuples of type $\langle n_s, n_d, n_p, N_f, s \rangle$.

C. Working Principle

The working principle of the protocol is simple. There is an algorithm that regulates the mobility of the nodes, that is, the local mobility heuristics used to create the disjointed paths. In parallel to the mobility task, there is another algorithm that makes the packet forwarding decision. Both algorithms work independently from one another but based on the same information (the flow of data packets) as feedback. This leads to a convergent solution for both mobility and packet forwarding.

The list of flows (F_n) that each node n is currently handling is periodically broadcast to all its immediate neighbors so that every node also maintains the list of flows being handled by its neighbors ($\mathcal{N}$). When receiving a data packet for forwarding, each node will either split the flow (if it hasn't been done yet by one of the upstream nodes) or attempt to move away from other nodes carrying the same flow on the *other* path.

The proposed approach is independent from the routing protocol being used. However, *dpath* does require for its operation the following capabilities from the routing service:

- *Enhanced Next-Hop Lookup:* That is, the possibility of looking up a *set* of best next-hop nodes for a given destination, as opposed to a single best-hop which is the conventional mode. Such information is often available in most routing protocols, but generally discarded and not included in the routing table.
- *Packet Forward Override:* This is necessary, for instance, for the splitting of a data flow, where a copy of the data packet (or interleave packets) are forwarded to both the best next-hop and the second best next-hop of a given destination.
- *Data Packet Modification/Encapsulation:* To reduce overhead, the state-information that is shared by *dpath* is minimized and is appended as part of the data packets themselves. For that, *dpath* requires the ability to annotate data packets with a flag indicating whether the flow has been split, together with the list of next hops, to which the packet will be sent. Therefore, the routing service must provide a facility for transmitting the annotated data packets.

In most cases, such capabilities can be easily implemented in a non-intrusive way that is fully compatible with the routing protocol. In our implementation, we make use of a cross-layer substrate [7], [8] that allows us to detect the flows of data in the network while also providing a routing component that enables the capabilities mentioned above.

IV. Algorithm Description

The packet forwarding and mobility algorithms work in parallel, based on the information about flows being exchanged over data packets. In this section we describe these algorithms and the mobility heuristics.

A. Packet Forwarding

The primary goal of the packet forwarding algorithm (PF) is to handle the splitting of the flow. The algorithm also maintains the flow separation by favoring (when possible) a next-hop that has not reported any participation in the current flow.

PF works as follows: when forwarding a packet P (a packet with source node n_s and destination node n_d), PF will check whether the flow to which the packet belongs, F_{n_s,n_d}, has already been split. In this case, PF will look for node n_1, the *best* next hop to n_d. If n_1 has not previously forwarded, from a different path (i.e., from a node different than the current node), a packet that belongs to F_{n_s,n_d}, then PF will forward P to n_1. In case n_1 has already forwarded a packet belonging to the flow F_{n_s,n_d}, PF will look for the next "best" hop to n_d. Otherwise, PF will drop P.

On the other hand, if the flow has not been previously split, PF will look for two different "best" next hops (n_1 and n_2) to use to forward the packet P towards n_d. If two such nodes are found, PF will mark the packet P to indicate that the flow has been split and then forward the packet to both n_1 and n_2. If two different routes to forward P could not be found, then P will be forwarded to the "best" next-hop available, and the flow is not marked as split.

B. Mobility Algorithm

The goal of the mobility algorithm is to "observe" the data flow and move the local node to create a disjointed path. Mobility decisions are local to each node, and are based on traffic and on information reported by other nodes. There is no global state required for node mobility, and information is shared only between immediate neighbors. Nodes are location aware, and the algorithm uses a node's position and that of its neighbors to decide where to move. Currently, the mobility algorithm only works for a single flow.

Figure 2 shows the initial topology for an NS-2 simulation of the dual-path algorithm in a 12 node network. In the simulation, a data flow starts in node '0', with node '11' as the destination. As illustrated, the packet-forwarding algorithm previously described takes advantage of disjointed paths already in place in the topology. The problem, however, is that without topology adaptation, the redundant data paths interfere with each other, thus potentially reducing (as opposed to improving) the robustness of the flow. Furthermore, there are two points in the flow that rely on a single node. The flow will be degraded (or fail to make it through) if nodes '2' or '3' are removed from this topology or fail to perform well.

The topology adaptation algorithm addresses this issue by allowing intermediate nodes to move autonomously to create a path from source to destination that is non-interfering (that is, node and edge disjoint, with no direct edges between intermediate nodes involved in different paths of the flow). As mentioned earlier, each node in the path periodically broadcasts its current position and information about the data flows

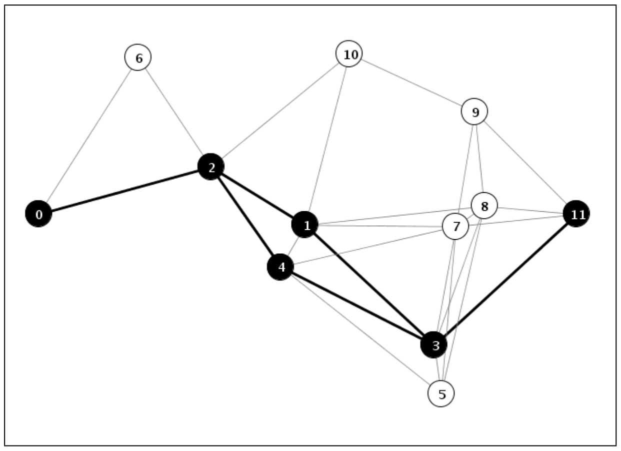

Fig. 2. Dual-path simulation, data flow from nodes 0 to 11.

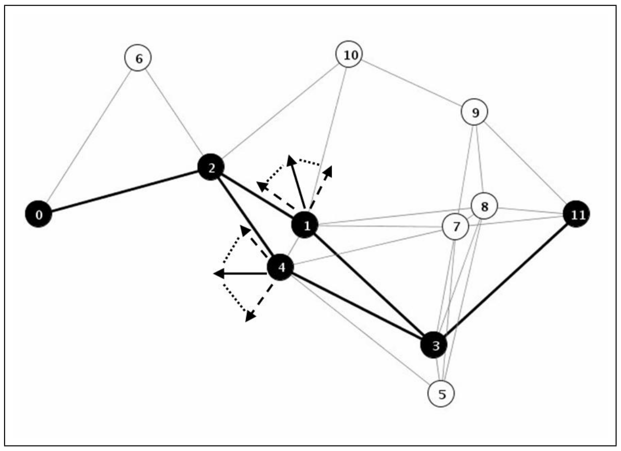

Fig. 3. Mobility vectors on some of the intermediate nodes.

that it is handling. Based on the broadcast information, each node can independently identify potential induced interference and decide how to move.

Figure 3 illustrates the adaptation process. The data flow from node '0' to node '11' is relayed by nodes '4' and '1'. These nodes determine that they are both interfering with each other in regards to the same data flow. Furthermore, they also identify that data is being relayed to them from the source in a non-split flow. This is an indication that node '2', the immediate upstream node, is either a common vertex in the path, or that there is a common edge leading to node '2'. Either way, the heuristic for node mobility in this case is to move away from interfering intermediate nodes, and towards upstream nodes that are unable to sub-divide the stream.

The composed mobility vectors are shown in Figure 3. Nodes '1' and '4' simultaneously move away from each other (to reduce interference) and towards their upstream neighbor (to create the conditions for a split from the source). The resulting direction is the composed vector illustrated in Figure 3. The other nodes involved in the path (with the exception of the source and target nodes) are also moving, but their mobility

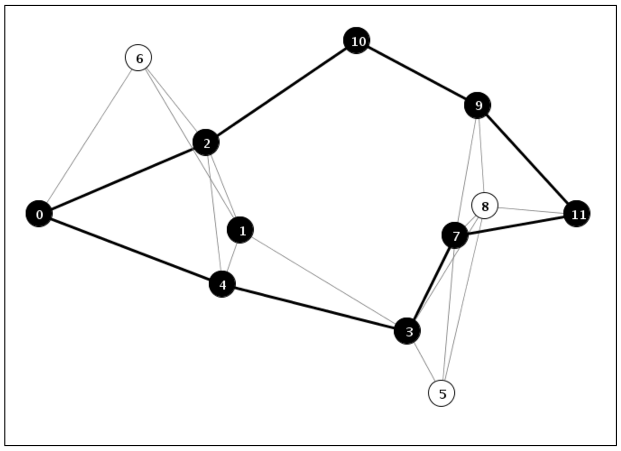

Fig. 4. Intermediate state with a dual-path stream created form the source node.

vectors are omitted here for simplicity.

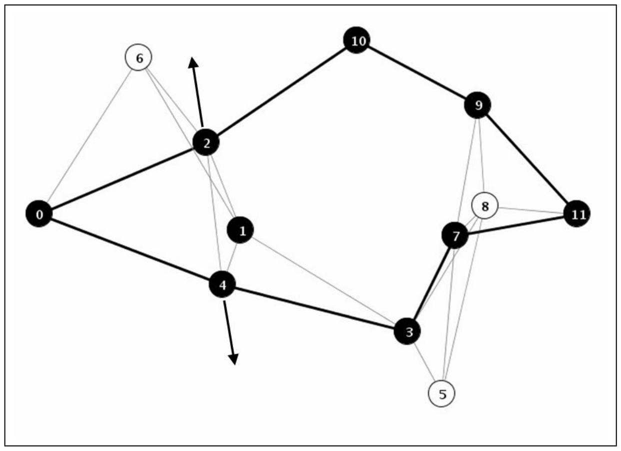

Fig. 5. Mobility vectors for nodes '2' and '4' to eliminate interference.

When node '0' finds two immediate paths to the target nodes, it immediately splits the flow, leading to the configuration shown in Figure 4. At this point, nodes '1' and '4' no longer need to move towards the source as they are each receiving a split of the flow. They are, however, still interfering with one another so they continue to move, but in the directions illustrated in Figure 5.

At the end of the process, the flow from node '0' to '11' is fully split at the source node, following separate, non-interfering paths, to later converge only at the target node (Figure 6), which is a stable state for the algorithm. At this point all intermediate nodes stop moving and data continues to flow.

Although all nodes become static again at the configuration illustrated in Figure 6, the dual-path algorithm continues to run, correcting its position again if a) interference is detected, or if b) a non-split flow coming from a node other than the source node is received by an intermediate node.

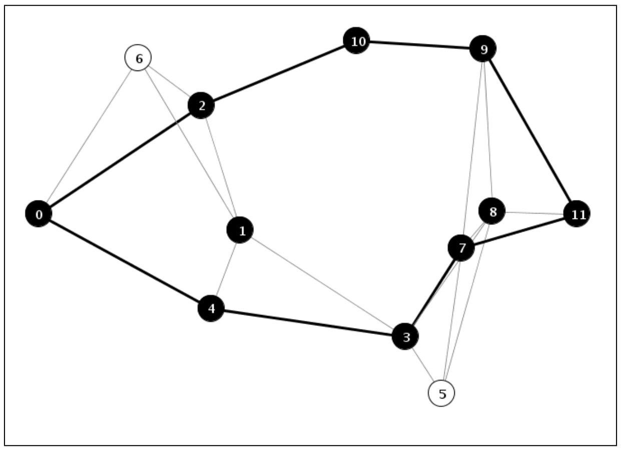

Fig. 6. Final Configuration with dual (disjoint) path from source to destination.

V. Conclusions and Future Work

With this preliminary study we have shown that it is possible to create non-interfering paths by modifying the topology of a MANET. However, our design only considers one concurrent data flow. Extending the algorithm to handle multiple data flows is our next step, although this is not a trivial task. An enhanced version of the algorithm requires better mobility models that can adapt to multiple flows.

Also, our current implementation attempts to minimize interference by manipulating the physical position of the nodes, and this is possible thanks to the fact that the nodes are mobile and mobility can be controlled. We need to explore the behavior of the algorithm in situations where node movement is just not possible. For these cases, modifying the transmission power of the wireless network device might help modify the topology to a desirable state.

References

[1] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Mobile Computing*, pp. 153–181, 1996.

[2] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing." in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications.*, 1999.

[3] T. Clausen and P. Jacquet, "RFC 3626: Optimized link state routing protocol (olsr)," Oct. 2003.

[4] S. Lee and M. Gerla, "Split multipath routing with maximally disjoint paths in ad hoc networks," in *IEEE International Conference on Communications, 2001. ICC 2001*, vol. 10, 2001.

[5] M. Marina and S. Das, "On-demand multipath distance vector routing in ad hoc networks," in *Proceedings of IEEE International Conference on Network Protocols (ICNP)*, vol. 1, 2001.

[6] S. Mao, Y. Hou, X. Cheng, H. Sherali, S. Midkiff, and Y. Zhang, "On routing for multiple description video over wireless ad hoc networks," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1063–1074, 2006.

[7] M. Carvalho, N. Suri, M. Arguedas, M. Rebeschini, and M. Breedy, "A Cross-Layer Communications Framework for Tactical Environments," in *Military Communications Conference, 2006. MILCOM 2006*, 2006, pp. 1–7.

[8] M. Carvalho, A. Granados, W. Naqvi, A. Brothers, J. Hanna, and K. Turck, "A cross-layer communications substrate for tactical Information Management Systems," in *IEEE Military Communications Conference, 2008. MILCOM 2008*, 2008, pp. 1–7.