Hive-Map Analysis of Mobile Ad-Hoc Networks

Greg Hansell[1]

Abstract— **This paper explores two publisher-subscriber-based routing protocols in a simulated mobile environment. The routing protocols are evaluated for reliability, latency, cost and fairness with low-level hardware considerations. The fairness evaluation highlights an unexpected issue with one of the routing protocols.**

I. INTRODUCTION

Mobile processors are ubiquitous: they exist in pockets, cars and drones! Imagine these components all working together in a decentralized way to communicate information of interest to one another. The sheer scale is intimidating, but there are communication paradigms that can combat this scale. One such communication paradigm is the publisher subscriber paradigm. This paper explores two publisher subscriber protocols for disseminating information in low-level mobile systems with limited communication abilities. The evaluation process uses a software tool called Hive-Map to facilitate the development and simulation of the two pub-sub protocols. This paper has two main objectives: (1) evaluate the two routing protocols with the Hive-Map framework, and (2) evaluate the Hive-Map framework with the two routing protocols. This seems circular, but Hive-Map is a budding framework that needs feedback. The paper is structured as follows. The first section provides background on pub-sub systems, mobile ad-hoc networks and the Hive-Map framework. The next section discusses the implementation of the two routing protocols. Then, the routing protocols are evaluated for reliability, latency, cost and fairness. Lastly, the conclusion covers the key takeaways of the results and future work.

II. BACKGROUND

Publisher subscriber systems (pub-sub systems) deal with information driven problems. A pub-sub system disseminates information to a group of participants with varying interests. New information is expressed as an Event. Interest in a certain type of event is expressed as a subscription. A pub-sub system delivers events to all interested subscribers. Topic-based pub-sub systems are a specific flavor of pub-sub where events have a topic and a message, and subscriptions express interest in certain topics. For example, if a subscription is interested in the topic "cat" then it will be notified of any events with "cat" as their topic. There are many other pub-sub flavors, but topic-based pub-sub systems are the focus of this paper because they are simple to reason about [4]. Pub-sub systems scale well with participants because events are a one-way conduit for information dispersal: a node could publish to countless subscribers or none and its behavior remains the same because that publisher is completely decoupled from all of the subscribers.

A node is an entry-point to a pub-sub system and provides two operations: publish and subscribe. Typically, nodes are spread across a network, and the pub-sub system is a network overlay that connects nodes. A Mobile Ad-hoc network (MANET) is a type of network that a pub-sub system can be built on [2]. In MANETs, a node experiences churn due to other nodes "moving" in-and-out of connection. MANET nodes do not lean on pre-existing network structures like wifi. To disseminate information past neighboring nodes, a pub-sub system needs a routing protocol that can deliver events to subscribers across the network. The routing protocol is the core component of any pub-sub system. A routing component sits on every node and decides what to do with local publishes, local subscriptions, and messages from neighboring nodes. This paper will explore two routing protocols: a naive gossip protocol and the hint-driven protocol.

Transceivers are one means of communication for nodes in a MANET environment. Nodes use transceivers to transmit and receive information from other nodes. Each transceiver has a communication range, and belongs to one node. If a node transmits bytes of data then only nodes within range will receive this data. If a node receives information from two nodes simultaneously then there is interference, and both messages received are lost; unless there is conflict logic. Two examples of transceivers are the nRF24L01 radio communication module and bluetooth. Bluetooth has a range of 100 meters; the nRF24L01 modules have a range of about 30-40 meters from personal experience, though the range is not in the specification because it is variable based on the hardware stack used [6].

A real world evaluation of pub-sub algorithms in a MANET environment is impractical: there can be countless nodes independently operating and physically moving! A pub-sub system, in a MANET environment, needs a simulation tool to debug and benchmark against other protocols. Luckily, there is Hive-Map. Hive-Map facilitates creating, simulating, and sharing pub-sub protocols. It breaks up the pub-sub problem into four modular interfaces: matcher, router, communicator and context. These interfaces can have different implementations. A matcher matches an event to interested subscriptions. The communicator sends and receives bytes from neighboring communicators. The router handles local publishes, local subscriptions and incoming messages from neighboring router components. These three components make up a node in the Hive-Map framework.

[1]Greg Hansell, Masters Student at Rensselaer Polytechnic Institute, gregjhansell@gmail.com

This paper implements the router interface for two protocols. External from the node is context, which provides necessary information to the node. For example, a node may need its location or a unique identifier to function; context ensures that the user will provide this information [5].

The Hive-Map framework offers a simulated radio transceiver from its python simulation library. This transceiver emulates key properties expected in a transceiver. First is the range property: messages only reach other transceivers in communication range. The simulated behavior is binary, and does not emulate a more complex radio propagation model. The second property is interference: if a transceiver receives multiple messages with intersecting time windows then all messages will be dropped. In addition there is some interference correction effort: if the intersecting transceiver messages can be heard by all sources then a random backoff is applied for the messages. The third property is the buffer size of a transceiver: messages received are put on a buffer waiting to be extracted by the transceiver user. Once the buffer reaches capacity, subsequent messages received will be dropped until there is enough space on the buffer.

III. Routing Protocols

This paper investigates two different routing protocols under the same system model. The focus is on lower-level systems like microcontrollers with limited memory and power. The two algorithms discussed are from the hint-driven protocol paper [1].

A. System Model

Messages are synchronous with an upper bound on delay, so a node can determine if other nodes are out of range. All messaging is broadcast-based: a node cannot directly message another node without sending that message to others in range. There is churn in the system when neighbors move in-and-out of range. Each node experiences message interference and limited memory. Communication is always bidirectional. All nodes can be uniquely identified and require a consistent sleep operation. This model is intended to emulate lower-level systems like microcontrollers. The consistent sleep operation is a challenging constraint for low quality processors with limited power consumption. It is a design consideration not to be taken lightly: the hint-router depends on this sleep consistency. Bidirectional communication is unrealistic, especially at the periphery of communication ranges, but it is necessary for the hint-driven protocol.

B. Gossip Protocol

The gossip protocol has one parameter that drives its behavior: forwarding probability. Forwarding probability determines whether a gossip router will forward a new non-local event after notifying its subscriptions. For example, if the forwarding probability is one, then a non-local event received will be forwarded unless the router has encountered that non-local event before. The gossip router always forwards a new local event. The forwarding probability is a dial: set it low when the network is dense and high if it is sparse. The gossip protocol causes a lot of unnecessary work: the system spreads the message in every direction in the hopes of blindly reaching a subscriber. On the other hand, the forwarding operation is simple and does not require consistent sleep, bidirectional communication or even a synchronous system model. This approach was chosen for its simplicity which makes it a good baseline to compare to. There are improvements to the gossip algorithm, but they will not be discussed for brevity [3].

C. Hint-Driven Protocol

The hint-driven protocol leverages a simple observation: a node U can be reached from node W if there is a set of intermediate nodes that get strictly closer to U on each hop. The protocol attempts to rebroadcast events only if it gets the events closer to interested subscriptions. The protocol has two types of messages: beacon messages and target messages. Beacon messages fire off after a globally known beacon interval, notifying neighbors of their local subscriptions. When a node receives a beacon, it logs the beacon's node interests and sets its hint value to 0. The hint value indicates how far away a neighbor is. Every node has a hint table that tracks its neighbors: when a neighboring node's beacon message is missed, the hint value is incremented for that node in the hint table. There is a max hint value. If this hint value is reached then that neighboring node is removed from the hint table. The larger the hint value for a once neighboring node, the further away that node is [1].

Target messages strive to get events closer to nodes with locally matching subscriptions. When the hint-routing protocol receives a locally published event, it broadcasts the event with a list of targets and a number of credits. These targets are nodes and their corresponding hint values. Each targeted node has local subscriptions interested in that event. Neighboring nodes receive this broadcast and only forward the message if (1) they have new targets to add, (2) the message has credits or (3) one of the targets is closer. The number of credits is decremented in the forwarded message if all other conditions are not satisfied. A delay is applied to the message before forwarding. The delay is proportional to the hint value of the closest target, but if only the credit condition is satisfied then the delay is proportional to the max hint value. A node will drop a pending message if it receives the pending message from a neighboring node because the node that forwarded the message earlier must be closer to one of the targets. The delay lowers the number of sent messages and keeps the focus on getting messages closer to targets. The credit system keeps messages alive even when neighboring nodes are not interested; the credit system increases reliability but also increases the number of messages sent [1].

The intended effect of the protocol is a wave of messages that ripples most violently towards targeted nodes. The actual effect is flawed because targets are clumped together into one message. If two pursued targets diverge then there is a chance

the closer target is exclusively pursued. For example, if two neighboring nodes receive a target message and one of them forwards the message earlier because it is closer to one of the targets then the other node will not forward the message even if it is closer to one of the other targets! Another issue arises when subscriptions change frequently, which would lead target messages to dead ends. Despite these cons, the algorithm does show promise in limiting the average number of messages sent per node while still holding reliability [1].

IV. Evaluation

The following subsections discuss the results of running the two aforementioned protocols in simulation. The focus of the results are on reliability, latency, cost, and fairness. Aside from benchmarking, the simulation tool will also be evaluated for its pitfalls and strengths.

A. Simulation

Parameter	Value	Parameter	Value
Field Area	1000x1000 m*m	Number of Node	100
Speed Range	10 - 20 m/s	Publishing Rate	0.5 msg/s
Simulation Duration	125 s	Number of Subscribers	10
Communication Range	150 m	Beacon Interval	5 s
Data Rate	2000 kbps	Hint Delay	0.1 s
Max Buffer Size	1024 bytes	Credit	[0, 1, 2]*
Forwarding Probability	[0.1, 0.3, 0.5, 0.7, 0.9,]*		

Fig. 1. simulation parameters

The simulation has 100 nodes in a 1000x1000 field. Each node moves in a random waypoint fashion. The simulation parameters are summarized in figure 1. The table is color coded. Orange cells are temporal and location parameters. Yellow cells are node specific parameters. Red cells are the simulated radio transceivers parameters. Green cells are the parameters for the hint-driven routing protocol. Lastly, blue cells are for the gossip routing protocol. The independent parameters adjusted were the hint-driven protocol's credits and the gossip protocol's forwarding probability.

A lot of the parameter values were taken from the original paper about the hint-driven protocol [1], but there are some differences. The first difference is the publishing rate. The hint-driven paper uses two publishers that publish a message every two seconds. My approach better simulates a system of publishes coming from a diverse set of nodes: the publishing rate specifies that a random non-subscribed node publishes an event every 2 seconds.

The second difference is in the simulated communicator used. The hint-driven paper used J-Sim. Which provides a full simulation of the 802.11 protocol stack and a detailed propagation model. This paper uses hive-map's simulated radio transceiver which is discussed in the background. The three radio-transceiver parameters were fixed for every test. In the NRF24 radio communication module, the data-rate is 2000 kbps [6]. To emulate this, a data-rate of 2000 is used in the experiments. If the data rate were smaller, then there would be more interference as message sends take longer to process. The max buffer size emulates small microcontrollers with limited memory capacities. The communication radius was suggested by the hint-driven protocol paper with one caveat: the other paper's transceivers had a probabilistic range from 100 to 200 meters, which was beyond the capabilities of Hive-Map's simulated radio transceiver, so a fixed communication range was set to 150 meters.

B. Reliability

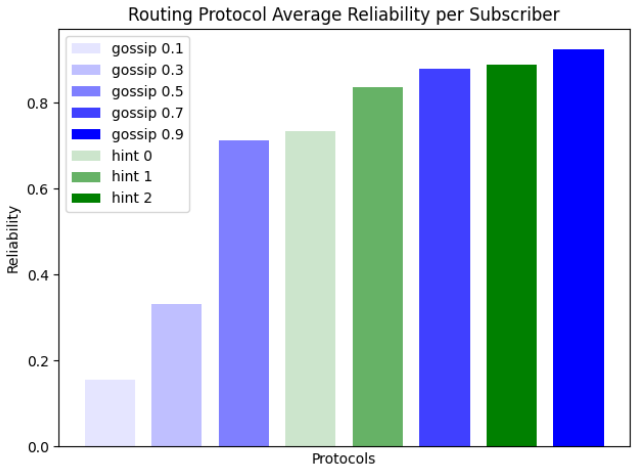

Fig. 2. simulated average reliability of the routing protocols

The reliability plot (figure 2) shows the two algorithms implemented and the average reliability per subscriber. There were 10 subscribers. The reliability is the number of events received by the subscriber over the total possible number of events received. It is impossible to have a reliability of 1 because there are network partitions. The reliability was a sanity check because this simulation was also performed in the hint-driven paper [1]. The gossip algorithm performed slightly better than in the hint-driven paper. The hint-driven protocol matched the original paper which is a good indicator that the protocol was implemented correctly and the simulation is behaving appropriately.

C. Latency

The latency plot (figure 3) shows the two algorithms and the average amount of time it takes an event to get from the publisher to the subscribers in seconds. If an event did not make it, then the event is not incorporated in the calculation. This makes the graph subtly deceiving: gossip 0.1 had the lowest latency, but also a really low reliability. This graph highlights the hint-driven protocol's use of credits. When a message uses a credit, it pauses for one second before rebroadcasting; this delay ensures that messages closer are sent first. Assuming the credits are effective: the more credits used, the longer the average latency will be. Both hint-1 and hint-2 performed similarly in both latency and reliability which indicates that there is a limit on how effective credits are.

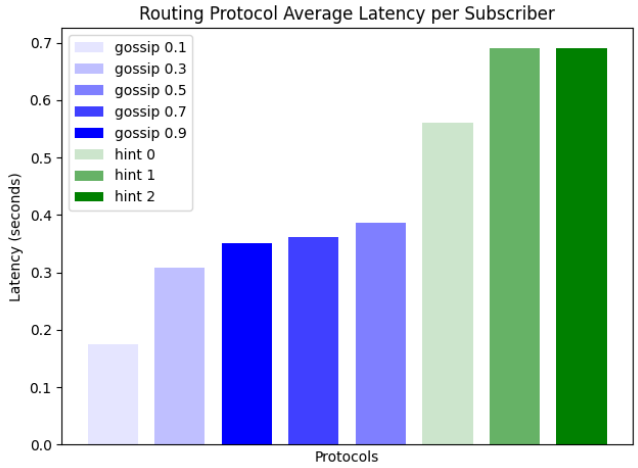

Fig. 3. simulated average latency of the routing protocols

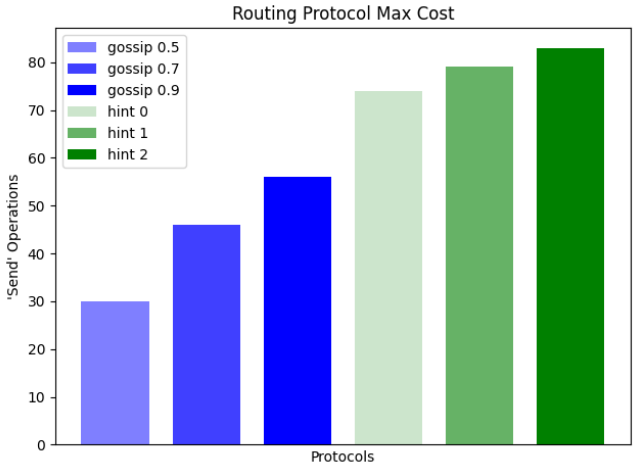

Fig. 5. simulated max cost of the routing protocols

D. Cost

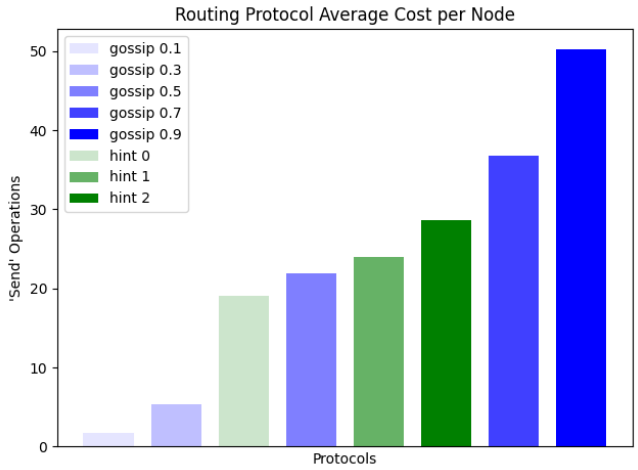

Fig. 4. simulated average cost of the routing protocols

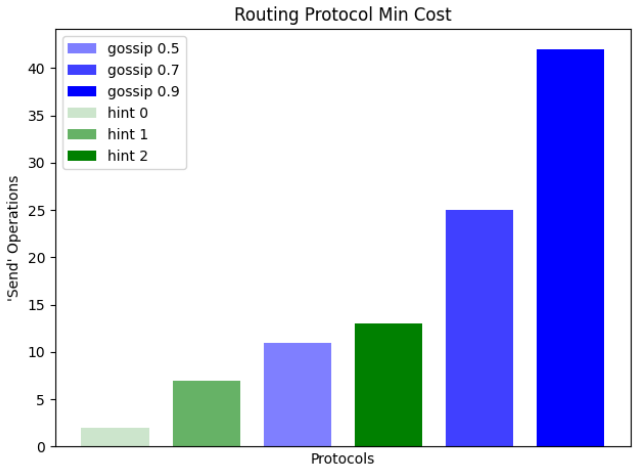

Fig. 6. simulated min cost of the routing protocols

Send operations are usually more expensive than receive operations in terms of power consumption and processor time. Thus send operations were used to determine the cost of the algorithm. Figure 4 highlights the hint protocol's cost-effectiveness. For example, hint-1 performs similarly to gossip-0.7 in regards to reliability (figure 2) and yet has nearly half as many send operations! The hint-routing paper measured network packets, and their results were even more stark [1]. I speculate their results were due to the simulation having twice as many publish operations coming from two publishers exclusively.

E. Fairness

For this benchmark, we show the max and min costs, figures 5 and 6 respectively. Cost is measured in the same way specified in the section prior; via the number of send operations. This comparison was chosen because max values can be hard limits for some systems. For example, a microcontroller could run out of power from sending too many messages and unexpectedly fail at critical operations. The results reveal that there are nodes in the hint routing algorithm that are doing more work than any of the gossiping algorithms! It is unclear from the data if these overworked nodes are playing a critical irreplaceable role, or if another able-bodied node could have performed the same send operation. Nonetheless, the hint-routing protocol has an uneven weighting on certain nodes.

F. Hive-Map

Selfishly, I wanted Hive-Map to succeed; and, because I wrote the software, I hold a biased view on its performance as a pub-sub framework. The simulated transceiver was helpful and behaved well even when used in a multi-process environment. An event-driven simulation tool (as opposed to cpu time) could have made the system more accurate, though the simulation yielded similar results to the hint paper [1]. One shortcoming of the simulated transceiver was scalability. The transceiver is written in python (a slow interpreted

language) and would not scale well to longer durations and larger number of nodes.

The Hive-Map components: context, router, matcher and communicator fit well together. This synergy allowed me to divide the problem and focus on the routing protocol. Two of the components need to be improved upon: context and matcher. Context is an important notion, but in the current implementation, context is pulled rather than pushed. For example, a location context is given to the transceivers, but if the location changes, the transceiver will not know unless it explicitly checks. The pull-paradigm of information does have performance gains in components that check context only when necessary, but some algorithms may depend on changes in context to trigger events.

The matcher needs to be improved upon as well. Local subscriptions are difficult to share between routers. The developer must serialize each individual interest and send it; then on the other end, these interests are deserialized one by one. Not to mention, comparing interests to prior interests in a look up table is cumbersome and difficult to maintain. These matcher problems are difficult to fix because the component must be flexible to diverse expressions of matching such as location-based matching; the entirety of the matching architecture may have to be reworked. As a side note, the matcher and communicator are not thread-safe. Thread-safety would have made some portions of the routing protocol simpler.

V. Conclusion

This paper compared two pub-sub protocols and focused on reliability, latency, cost, and fairness. The gossip protocol had adjustable reliability, low latency, and fair workloads across components; these benefits came at the price of send operations, and in a low-level setting a large number of send operations could kill battery life. The evaluation section revealed new insight into the hint-driven routing protocol. The hint-driven protocol showed good reliability and a low number of average sends, but it struggled with fairness and latency. The lower number of sends is true collectively, but not at the node level. The node that performed the most send operations was running the hint driven protocol with 2 credits (figure 5). This has huge ramifications for hardware components. Designers are usually more concerned about max load than they are average load. Future work is needed to determine how critical these overworked nodes are. The Hive-Map framework was sufficient enough to simulate the system. Most of its limitations were caused by the difficulty to share and serialize interests across nodes.

References

[1] Baldoni, R., Beraldi, R., Cugola, G., Migliavacca, M., Querzoni, L.: “Structure-less Content- Based Routing in Mobile Ad Hoc Networks”. (2005)

[2] Baldoni, R., Querzoni, L., Virgillito, A.: “Distributed Event Routing in Publish/Subscribe Communication Systems: a Survey”. (2009)

[3] Costa, P., Migliavacca, M., Picco, G., Cugola, G.: “Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation”. (2009)

[4] Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: “The Many Faces of Publish/Subscribe”. ACM Computing Surveys 35 (2003) 114–131

[5] Hansell, G.: “Hive-Map”. In: https://github.com/gregjhansell97/hive-map (2020)

[6] Nordic Semiconductor. “Single chip 2.4 GHz Transceiver nRF24L01”.