Article

Not peer-reviewed version

Cross-Layer Optimization for Enhanced IoT Connectivity: A Novel Routing Protocol for Opportunistic Networks

Ayman Khalil * and Besma Zeddini

Posted Date: 4 April 2024

doi: 10.20944/preprints202404.0342.v1

Keywords: cross-layer; message delivery; probabilistic routing; history-based algorithm

Article

Cross-Layer Optimization for Enhanced IoT Connectivity: A Novel Routing Protocol for Opportunistic Networks

Ayman Khalil [1, ‡,*] and Besma Zeddini [2,‡]

[1] School of Business Lebanese American University Beirut, Lebanon
[2] SATIE Laboratory CNRS – UMR 8029; CY Tech, CY Cergy Paris University; besma.zeddini@cyu.fr
[‡] These authors contributed equally to this work.
* Correspondence: ayman.khalil02@lau.edu.lb

Abstract: Opportunistic networks, an evolution of mobile Ad Hoc networks (MANETs), offer decentralized communication without relying on preinstalled infrastructure, enabling nodes to route packets through different mobile nodes dynamically. However, due to the absence of complete paths and rapidly changing connectivity, routing in opportunistic networks presents unique challenges. This paper proposes a novel probabilistic routing model for opportunistic networks, leveraging nodes' meeting probabilities to route packets towards their destinations. The model dynamically builds routes based on the likelihood of encountering the destination node, considering factors such as last meeting time and acknowledgment tables to manage network overload. Additionally, an efficient message detection scheme is introduced to alleviate high overhead by selectively deleting messages from buffers during congestion. Furthermore, the proposed model incorporates cross-layer optimization techniques, integrating optimization strategies across multiple protocol layers to maximize energy efficiency, adaptability, and message delivery reliability. Through extensive simulations, the effectiveness of the proposed model is demonstrated, showing improved message delivery probability while maintaining reasonable overhead and latency. This research contributes to the advancement of opportunistic networks, particularly in enhancing connectivity and efficiency for Internet of Things (IoT) applications deployed in challenging environments.

Keywords: cross-layer**;** message delivery; probabilistic routing; history-based algorithm

1. Introduction

Opportunistic networks represent a paradigm shift in communication, offering decentralized and resilient connectivity in scenarios where traditional infrastructure-based networks may be impractical or unavailable [1]. These networks, born from the evolution of mobile Ad Hoc networks (MANETs), enable communication between nodes without relying on pre-established infrastructure, allowing packets to be routed dynamically through various mobile nodes in the network. However, the inherently sporadic and unpredictable nature of opportunistic networks poses significant challenges for routing and message delivery. In opportunistic networks, the absence of complete end-to-end paths between source and destination nodes requires routing protocols to adapt dynamically to changing network conditions. Traditional routing approaches designed for Internet routing are ill-suited for such environments, necessitating the development of novel routing models tailored to the unique characteristics of opportunistic networks. Additionally, the resource-constrained nature of mobile devices in these networks imposes constraints on energy consumption, requiring energy-efficient routing strategies to prolong network lifespan [1,2].

This paper proposes a novel cross-layer probabilistic routing model for opportunistic networks, designed to address the challenges of dynamic routing and energy efficiency. The proposed model leverages nodes' meeting probabilities to route packets towards their destinations, dynamically adjusting routes based on encounter history and network conditions. Furthermore, the model incorporates techniques integrating strategies across multiple protocol layers to maximize energy efficiency, adaptability, and message delivery reliability.

This paper presents a comprehensive exploration of leveraging opportunistic networks for enhancing Internet of Things (IoT) communication in section 2. The proposed solution is presented in Section 3, where we introduce a novel framework employing probabilistic routing algorithms and efficient resource utilization mechanisms to optimize message delivery and address connectivity challenges in dynamic environments. Subsequently, Section 4 details the empirical results and performance evaluations of our proposed solution, presenting significant improvements in several performance metrics. Finally, in Section 5, the paper concludes by summarizing the findings, emphasizing the potential of opportunistic networks to enhance IoT communication.

2. IoT and Opportunistic Networks – State of the Art

IoT systems involve interconnected smart devices equipped with sensors and communication modules. Recent research emphasizes the importance of efficient data routing, scalability, and interoperability in IoT architectures [1]. Opportunistic Networks operate in intermittently connected environments, relying on node mobility and opportunistic contacts for communication. Recent studies focus on routing protocols, mobility models, and resource optimization techniques to enhance communication reliability and efficiency [2]. The integration of IoT with Opportunistic Networks enables applications in disaster management, wildlife monitoring, and rural connectivity. Advanced routing protocols play a crucial role in facilitating data dissemination, energy-efficient routing, and adaptive forwarding strategies [3].

2.1. Applications

- Disaster Management: Advanced routing protocols in IoT-enabled opportunistic networks facilitate resilient data dissemination and coordination in disaster scenarios. Recent research emphasizes the development of delay-tolerant routing algorithms and message ferrying strategies to ensure timely delivery of critical information [4].
- Wildlife Monitoring: Routing protocols optimized for energy efficiency and adaptability enhance data collection and transmission in wildlife monitoring applications. Recent studies explore opportunistic routing schemes, content-centric protocols, and cooperative forwarding strategies to extend network coverage and reduce communication overhead [5].
- Rural Connectivity: Advanced routing protocols address connectivity challenges in IoT deployments in rural areas. Research focuses on geographic routing, social-aware forwarding, and multi-path selection techniques to optimize message delivery and mitigate the impact of network partitions [6].

2.2. Challenges

- Intermittent Connectivity: Routing in opportunistic networks with intermittent connectivity requires novel approaches to handle network disruptions and message delivery delays. Recent research emphasizes the design of store-carry-and-forward mechanisms, encounter-based routing, and epidemic protocols to adapt to changing network conditions [7].
- Dynamic Topology: Opportunistic networks exhibit dynamic topology due to node mobility and intermittent contacts. Advanced routing protocols incorporate mobility prediction, context-aware routing, and network coding techniques to improve routing efficiency and adaptability [8].
- Resource Constraints: IoT devices operating in opportunistic networks have limited resources, including battery power and bandwidth. Routing protocols must optimize resource utilization through energy-aware routing, data aggregation, and adaptive transmission strategies [9].
- Content-Centric Networking: Future research directions include the exploration of content-centric networking paradigms in IoT-enabled opportunistic networks. Content-based routing, caching, and dissemination strategies offer opportunities to enhance data delivery efficiency and support diverse application requirements [10].

- Cross-Layer Optimization: Leveraging cross-layer optimization techniques to enhance routing performance and resource utilization in IoT-enabled opportunistic networks. Integration of network, transport, and application layer functionalities enables holistic optimization and adaptation to dynamic network conditions [11].
- Edge Computing Integration: Integration of edge computing capabilities with opportunistic networks to support localized data processing and decision-making. Edge-aware routing protocols and distributed computing frameworks enable efficient utilization of edge resources and reduce reliance on centralized infrastructure [12].

2.3. Routing in Opportunistic Networks

Recent research has witnessed a surge in the development of novel routing protocols tailored to the challenges of opportunistic networks. Zhang et al. introduced CARP, a context-aware routing protocol that utilizes contextual information such as node mobility patterns and encounter history to make routing decisions dynamically [13]. Li et al. proposed a hybrid routing protocol, combining epidemic routing with social-based forwarding mechanisms, which exploit social relationships between nodes to enhance message dissemination efficiency [14]. Additionally, Chen et al. presented MARP, a mobility-aware routing protocol that adapts routing strategies based on predictive mobility models and encounter probabilities, ensuring robustness against node mobility and network dynamics [15]. Wang et al. introduced ACO-RP, a bio-inspired routing protocol inspired by ant colony optimization principles, which demonstrated superior scalability and message delivery efficiency [16]. Moreover, Yang et al. proposed RUPON, leveraging reinforcement learning techniques to optimize routing decisions based on network dynamics and historical data, thereby achieving adaptive and efficient routing in opportunistic networks [17]. Similarly, Liu et al. introduced PRON, a policy routing protocol based on deep reinforcement learning, which learns optimal routing policies from historical experiences and network conditions to achieve improved message delivery performance [18]. These innovative routing protocols employ advanced techniques such as context awareness, social relationships, mobility prediction, and bio-inspired algorithms to overcome the challenges of intermittently connected environments, paving the way for efficient and adaptive communication in opportunistic networks.

2.4. Energy Efficiency in Opportunistic Networks and IoT

In opportunistic networks, energy-efficient communication is essential due to the sporadic nature of network connectivity and the limited battery life of mobile devices. Researchers have proposed various techniques to optimize energy consumption while maintaining communication reliability. For instance, Zhao et al. introduced an energy-aware routing protocol for opportunistic networks, which dynamically selects routes considering both energy levels and transmission opportunities [19]. Similarly, Li et al. proposed an energy-efficient data dissemination scheme based on social network analysis, where nodes with higher energy reserves are prioritized for message forwarding [20]. These studies highlight the importance of energy-awareness in routing and data dissemination strategies to prolong network lifetime in opportunistic environments.

In IoT deployments, energy efficiency is crucial for prolonging device lifespan and reducing maintenance overhead. Recent research has focused on optimizing energy consumption at both the device and network levels. For example, Kim et al. proposed an energy-efficient IoT architecture leveraging edge computing and data aggregation techniques to minimize data transmission and processing overhead [21]. Additionally, Zhang et al. introduced an energy-efficient MAC protocol for IoT networks, which adapts transmission parameters based on channel conditions to reduce energy consumption during data transmission [22]. These studies demonstrate the significance of energy-efficient protocols and architectures in enabling sustainable IoT deployments.

Recent advancements in energy efficiency emphasize cross-layer designs that integrate optimization techniques across different protocol layers. For instance, Liu et al. proposed a cross-layer optimization framework for IoT networks, which jointly optimizes routing, MAC, and physical layer parameters to minimize energy consumption while ensuring QoS requirements [23]. Similarly, Wang et al. introduced a cross-layer energy management scheme for opportunistic networks, which dynamically adjusts routing and transmission parameters based on energy availability and network conditions [24]. These studies highlight the potential of cross-layer approaches in maximizing energy efficiency in both opportunistic networks and IoT deployments.

2.4. Paper Contribution

Our contribution lies in addressing the unique challenges posed by opportunistic networks, which represent an evolution of MANETs where communication is independent of preinstalled infrastructure, relying instead on routing packets through various mobile nodes. In these networks, the absence of a complete path from source to destination and rapidly changing or breaking paths necessitate nodes to store and forward data opportunistically when connectivity permits. This differs fundamentally from conventional Internet routing, as routes are dynamically constructed, and any reachable node can serve as a potential relay towards the destination. To tackle these challenges, we propose a novel probabilistic routing model that leverages nodes' encounter probabilities to guide packet forwarding. Our approach capitalizes on the likelihood of a node encountering the destination based on past interactions, alongside an acknowledgment table mechanism to alleviate network overload. Additionally, to mitigate high overhead, we introduce an efficient message detection scheme, facilitating the removal of selected messages from buffers during congestion. Through extensive large-scale simulations, our model demonstrates remarkable improvements in message delivery probability, while maintaining reasonable overhead and latency, thus offering a promising solution to the complexities of opportunistic networks. The proposed solution integrates functionalities across different layers of the network protocol stack:

- Physical and Data Link Layers:

While not explicitly mentioned, the opportunistic nature of the network implies that the solution likely accounts for varying physical and link-layer conditions, such as node mobility and intermittent connectivity. These considerations are fundamental to the operation of opportunistic networks.

- Network Layer:

The proposed probabilistic routing model optimizes routing decisions based on the likelihood of nodes encountering the destination. This involves considering node mobility patterns and historical meeting times, which are characteristic of the network layer. By routing packets based on nodes' meeting probabilities, the solution adapts to the dynamic and unpredictable nature of opportunistic networks, thereby enhancing routing efficiency.

- Transport Layer:

The model relies on an acknowledgment table to release network overload. Acknowledgment mechanisms typically operate at the transport layer to ensure reliable delivery of data. By managing network congestion using an acknowledgment table, the solution addresses issues related to network overload, contributing to improved performance and resource utilization.

- Integration of Functionalities:

The solution integrates functionalities across multiple layers of the protocol stack to optimize routing in opportunistic networks. By leveraging probabilistic routing at the network layer and acknowledgment mechanisms at the transport layer, the solution addresses challenges such as node mobility, intermittent connectivity, and network congestion.

- Optimization Across Layers:

The proposed solution optimizes performance across multiple layers by dynamically adapting routing decisions based on node encounter probabilities, managing network congestion using acknowledgment mechanisms, and implementing efficient message detection schemes to alleviate high overhead. These optimizations contribute to improving message delivery probability, reducing overhead, and minimizing latency in opportunistic networks.

3. Proposed Model

In a real opportunistic network involving human activity where mobile nodes are carried by humans, the mobility is not entirely random. Human activity introduces patterns in mobility, which exhibit characteristic time intervals depending on the specific activities being performed. This introduces a cross-layer consideration where the routing approach is influenced by both mobility patterns and communication protocols. The proposed routing approach incorporates the probability of a current node encountering the destination node based on their last meeting time. Additionally, the model integrates an acknowledgment table to facilitate the removal of acknowledged messages from the network, effectively managing data across layers. To address potential high overhead, the protocol employs efficient mechanisms for detecting and managing message congestion, ensuring optimal performance in cross-layer communication scenarios. The routing protocol requires the presence of the two tables presented in Figure 1 in the memory of each mobile node. The "Acked Messages Table" is utilized to store the IDs of all acknowledged messages, along with their acknowledgment time and time-to-live (TTL). The flowchart depicted in Figure 2 illustrates the cyclical process managed by this table.

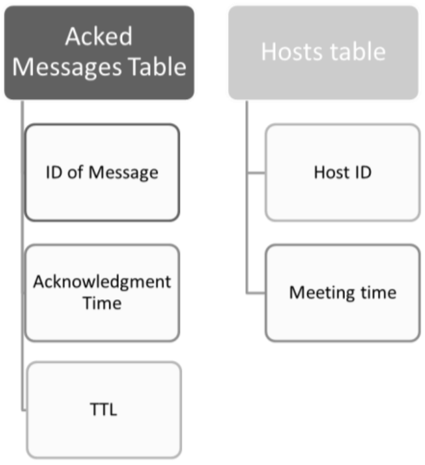


Figure 1. Schematic of two required tables.

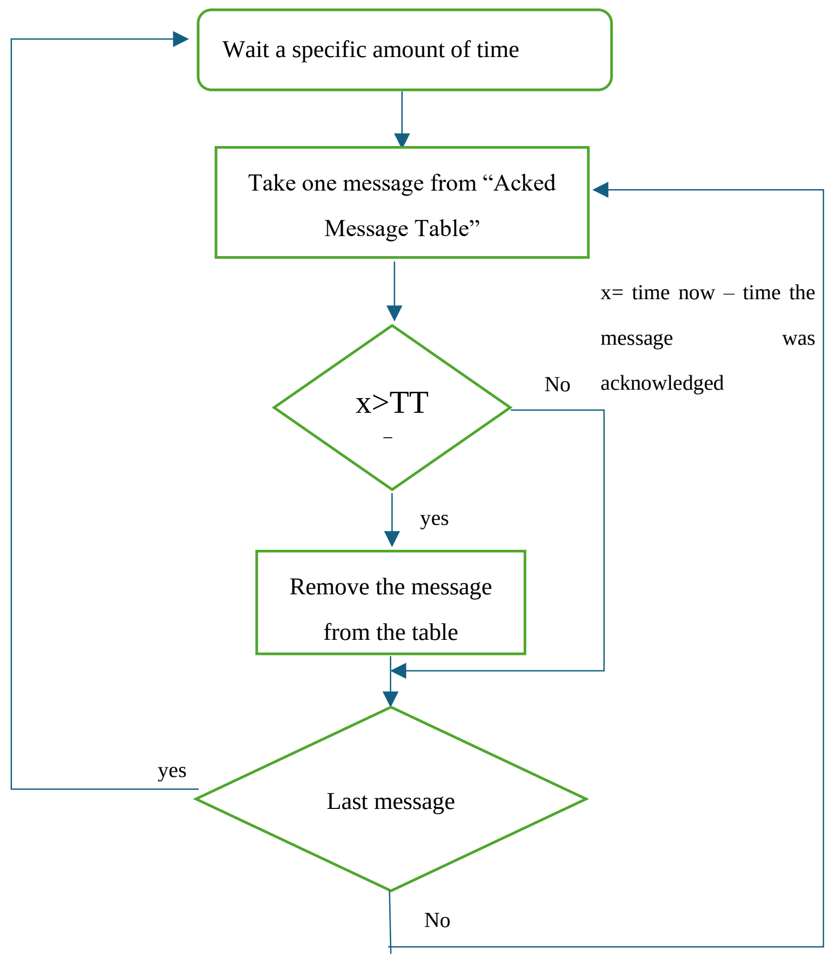
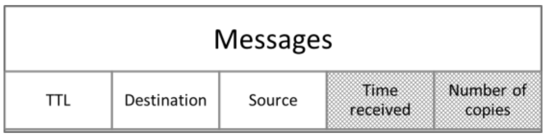

Wait a specific amount of time
Take one message from "Acked Message Table"
x= time now – time the message was acknowledged
x>TT
No
yes
Remove the message from the table
yes
Last message
No

Figure 2. Flowchart process of the Acked message table.

To mitigate table size inflation, a routine examination of the table verifies whether the duration since message acknowledgment exceeds or equals the TTL. If affirmed, the message is expunged from the table, consequently removing all duplicates across the network due to TTL expiration. Additionally, the "Hosts table" is implemented to store encountered node IDs and their respective last encounter times by the current node.

An additional requirement entails augmenting each message with an integer header titled "Number of Copies" and a new entry denoted as "Time Received." The "Number of Copies" header regulates message proliferation within the network, thus curbing overhead and enhancing delivery probability. Figure 3 illustrates the message overhead, with the dashed sections indicating the newly incorporated fields.



Messages				
TTL	Destination	Source	Time received	Number of copies

Figure 3. Message overhead.

Model Phases

The proposed model consists of 5 phases:

- **Phase 1: Initialization**

Upon establishing a connection between two nodes, this phase initiates with the objective of purging all acknowledged messages from the buffers of both connecting nodes while updating their encounter timestamps. Deleting acknowledged messages serves to mitigate buffer congestion, making room for newly received messages (as detailed in phase 5), and prevents redundant message transmission and processing, thereby conserving transmitting and processing resources. As depicted in Algorithm 1, upon connection establishment, both nodes exchange their "Acknowledged Messages Tables," incorporating delivered messages from the received table into their own and subsequently removing these delivered messages from their respective buffers. It is noteworthy that preserving the IDs of delivered messages in a table and exchanging this table during connection establishment proves more efficient than flooding the network with acknowledgment messages for each delivered message. This approach reduces overhead, mitigates buffer overflow, and conserves energy. Furthermore, leveraging the acknowledgment table allows the sender node to ascertain whether the message reaches its destination. Following the deletion of acknowledged messages, both nodes check if the other node is listed in their "Hosts Table." If affirmative, they update their encounter timestamps; otherwise, the node is added to the table along with its encounter time. Upon completion of this phase, the routing protocol proceeds to the subsequent phase.

Algorithm 1: Initialization Phase

```
1. DeleteAcknowledgedMessages(buffer, ackedMessagesTable):
2.   For each message in buffer:
3.       If message.ID is in ackedMessagesTable:
4.           Remove message from buffer
5. UpdateMeetingTime(node):
6. If otherNode is in node.hostsTable:
7.       UpdateMeetingTime(node, otherNode)
8. Else:
9.       AddNodeToHostsTable(node, otherNode)
10. ExchangeAckedMessagesTables(connection):
11. For each node in connection:
12.       AddDeliveredMessagesToTable(node)
13.       SendAckedMessagesTable(node, connection)
14. UpdateHostsTable(connection):
15. For each node in connection:
16.       If node is not in node1.hostsTable:
17.           AddNodeToHostsTable(node1, node)
18. AddDeliveredMessagesToTable(node):
19. For each message in node.receivedAckedMessagesTable:
20.       If message.ID is not in node.ackedMessagesTable:
21.           Add message.ID to node.ackedMessagesTable
```

- **Phase 2: Exchange Deliverables**

The primary objective of this phase is to prioritize the exchange of deliverable messages, those intended for the directly connected node. By ensuring these messages are transmitted first, it optimizes the delivery process and enhances the probability of successful message delivery. This strategy is particularly crucial as the buffer contains both deliverable messages and those intended for other nodes within the network. By prioritizing deliverable messages, nodes with lower power can efficiently function as direct delivery nodes, as depicted in phase 5. Furthermore, should the connection between nodes abruptly close due to node speed or other malfunctions, prioritizing deliverable messages ensures their timely delivery, thereby increasing overall delivery reliability. In this process, both nodes accumulate deliverable messages and exchange them accordingly. Subsequently, each node updates its "Acked Messages Table" with the newly delivered messages. If the node's energy level exceeds the predefined threshold, the routing protocol proceeds to the subsequent phase, which outlines the handling of other messages within the buffer. Conversely, if the node's energy falls below the threshold, the connection is terminated to conserve energy, and the node operates as a direct delivery node. This approach is illustrated in Algorithm 2.

Algorithm 2: Exchange Deliverables

```
1. PrioritizeDeliverableMessagesPhase():
2.     AccumulateDeliverableMessages()
3.     ExchangeDeliverableMessages()
4.     UpdateAckedMessagesTable()
5.     If EnergyExceedsThreshold():
6.         MoveToNextPhase()
7.     Else:
8.         CloseConnectionToPreserveEnergy()
9. AccumulateDeliverableMessages():
10.    For each message in buffer:
11.        If message.Destination is connected node:
12.            Add message to deliverableMessagesList
13. ExchangeDeliverableMessages():
14.    Send deliverableMessagesList to connected node
15.    Receive deliverableMessagesList from connected node
16. UpdateAckedMessagesTable():
17.    For each message in receivedMessagesList:
18.        If message not in ackedMessagesTable:
19.            Add message to ackedMessagesTable
20. EnergyExceedsThreshold():
21.    If node.energy > thresholdEnergy:
22.        Return true
23.    Else:
24.        Return false
25. MoveToNextPhase():
26.    Proceed to next phase of routing protocol
27. CloseConnectionToPreserveEnergy():
28.    Close connection with connected node
29.    Operate as direct delivery node
```

- **Phase 3: Procedures for Non-Deliverable Messages**

This phase serves as the fundamental component of the newly devised routing protocol. It is noteworthy that, at this juncture, the message buffer exclusively comprises non-deliverable messages, denoted as "messages with final destinations other than the connected node." The primary objective of this phase is to gather the messages earmarked for transmission to the other node within a temporary output buffer. The selection of messages to populate the output buffer hinges upon specific criteria delineated in Algorithm 3. The initial block, denoted "for each message in the buffer," signifies the possibility of implementing this phase wherein all messages are processed concurrently. However, the simulator's implementation does not involve multithreading; rather, it sequentially evaluates each message. The processing sequence initiates by addressing scenarios where at least one of the nodes fails to reach the destination. In this context, let x denote the time elapsed since the current node encountered the destination node, and y represent the duration since the other node met the destination node. If the current node hasn't encountered the message's intended destination while the other node has, the message is replicated into the output buffer. Conversely, if the current node has reached the destination while the other node has not, the message is excluded from the output buffer. However, in scenarios where both nodes have reached the destination, if the current node's meeting time with the destination precedes that of the other node, the message is omitted from the output buffer. Conversely, if the current node's meeting time is later than that of the other node, the message is copied into the output buffer. This decision-making process is illustrated in the flowchart, particularly in the case where the condition x < y is tested.

Algorithm 3: Non-Deliverable messages

```
1. CoreRoutingPhase():
2.     InitializeOutputBuffer()
3.     For each message in buffer:
4.         DetermineMessageDestination()
5.         ProcessMessageForOutputBuffer()
6. InitializeOutputBuffer():
7.     Create an empty temporary output buffer
8. DetermineMessageDestination():
9.     Identify whether message's destination is the connected node or another node in the network
10. ProcessMessageForOutputBuffer():
11.     If message's destination is another node:
12.         DetermineNodeMeetingTimes()
13.         CopyMessageToOutputBufferBasedOnCriteria()
14. DetermineNodeMeetingTimes():
15.     Calculate time elapsed since current node met destination (x) and since other node met destination (y)
16. CopyMessageToOutputBufferBasedOnCriteria():
17.     If current node didn't meet destination while other node did:
18.         Copy message to output buffer
19.     Else if current node met destination while other node didn't:
20.         Exclude message from output buffer
21.     Else:
22.         If x < y:
23.             Exclude message from output buffer
24.         Else:
25.             Copy message to output buffer
```

Phase 4: Send messages accepted by the receiver

In this synchronized process, the current phase operates at the sender side, while the subsequent phase executes at the receiver side. Initially, the sender gathers the metadata of all messages stored in the output buffer into a single message, which is then transmitted to the receiver. This metadata comprises the message ID and its size. Subsequently, the sender awaits a response from the receiver for each message. Upon receiving the response, the sender determines whether to proceed with sending the message or to remove it from the output buffer based on the received reply. It's essential to note that the receiver's responses are contingent upon specific criteria elucidated in the fifth phase of the protocol. As depicted in Algorithm 4, the sender remains in a waiting state until a reply is received from the receiver. Each reply message issued by the receiver contains either "RCV_OK" or "DENIED_LOW_RESOURCES," along with the corresponding message ID for which the reply is provided. If the receiver responds with "RCV_OK," the sender dispatches the message from the output buffer whose ID matches that in the reply message and then returns to the waiting state. Conversely, if the reply is "DENIED_LOW_RESOURCES," signifying insufficient resources at the receiver, the connection between the sender and receiver is terminated.

Algorithm 4: Messages accepted by the receiver

```
1. SynchronizedDataExchangePhase():
2.      SenderSideProcessing()
3.      ReceiverSideProcessing()
4. SenderSideProcessing():
5.      GatherMetadataOfMessages()
6.      CreateMetadataMessage()
7.      TransmitMetadataMessageToReceiver()
8.      WaitForResponseFromReceiver()
9.      ProcessReceiverResponse()
10. ReceiverSideProcessing():
11.     WaitForMetadataMessage()
12.     ExtractMetadataFromMessage()
13.     SendResponseForEachMessage()
14. GatherMetadataOfMessages():
15.     For each message in output buffer:
16.         Extract message ID and size
17.         Store metadata in temporary structure
18. CreateMetadataMessage():
19.     Construct a message containing metadata of all messages in output buffer
20. TransmitMetadataMessageToReceiver():
21.     Send metadata message to receiver
22. WaitForResponseFromReceiver():
23.     Wait for reply from receiver
24. ProcessReceiverResponse():
25.     If response is RCV_OK:
26.         SendMessageFromOutputBuffer()
27.     Else if response is DENIED_LOW_RESOURCES:
28.         CloseConnectionWithReceiver()
29. WaitForMetadataMessage():
30.     Wait for metadata message from sender
31. ExtractMetadataFromMessage():
32.     Extract message IDs and sizes from metadata message
33. SendResponseForEachMessage():
34.     For each message metadata:
```

```
35.         EvaluateResponseCriteria()
36.         Send appropriate response to sender
37. EvaluateResponseCriteria():
38.     If sufficient resources available:
39.         Respond with RCV_OK
40.     Else:
41.         Respond with DENIED_LOW_RESOURCES
```

Phase 5: Message acceptance criteria

During this phase, occurring at the receiver side subsequent to receiving message metadata dispatched by the sender in the fourth phase, a series of actions ensue. Should the energy level of the receiver's node fall below a predefined minimum threshold, a "DENIED_LOW_RESOURCES" response is relayed to the sender, leading to termination of the connection. Conversely, if the energy level is deemed sufficient, the receiver adheres to specific criteria to determine the messages to be accommodated within its buffer. As depicted in Algorithm 5, if the receiver possesses adequate energy and the message is absent in the buffer, several checks are performed, primarily concerning the message's size. Should the message size exceed the buffer capacity, it is rejected. Conversely, if the available free buffer space meets or exceeds the message size, the message is accepted. In scenarios where the free buffer space falls short of the message size, the receiver systematically removes the oldest messages from the buffer until sufficient space is available to accommodate the incoming message. Upon acceptance of the message, the receiver dispatches a "RCV_OK" packet to the sender, containing the message ID, confirming successful reception and readiness for subsequent transmission.

Algorithm 5: Messages acceptance criteria

```
1. ReceiverSideProcessing():
2.     ReceiveMetadataFromSender()
3.     CheckNodeEnergy()
4.     If EnergyBelowThreshold():
5.         SendDenialResponse()
6.     Else:
7.         ProcessIncomingMessages()
8. ProcessIncomingMessages():
9.     For each message metadata received:
10.         CheckMessageAcceptanceCriteria()
11.         If MessageAccepted():
12.             AcceptMessageAndSendConfirmation()
13.         Else:
14.             ContinueToNextMessage()
15. CheckMessageAcceptanceCriteria():
16.     If MessageInBuffer():
17.         Return false
18.     If MessageSizeExceedsBufferCapacity():
19.         Return false
20.     If FreeBufferSpaceSufficient():
21.         Return true
```

```
22.     Else:
23.         DeleteOldestMessagesUntilSpaceAvailable()
24.         **Return** true
25. DeleteOldestMessagesUntilSpaceAvailable():
26.     While FreeBufferSpaceLessThanMessageSize():
27.         DeleteOldestMessageFromBuffer()
28. AcceptMessageAndSendConfirmation():
29.     SendResponseToSender("RCV_OK", MessageID)
30. ReceiveMetadataFromSender():
31.     Receive metadata message from sender
32. CheckNodeEnergy():
33.     Check energy level of receiver's node
34. EnergyBelowThreshold():
35.     Determine if energy level falls below minimum threshold
36. SendDenialResponse():
37.     Send response to sender indicating insufficient resources ("DENIED_LOW_RESOURCES")
```

4. Performance Evaluation

The simulator employed in the proposed research is ONE (Opportunistic Network Environment) simulator, distinguishing itself from other Delay-Tolerant Networking (DTN) simulators [25,26]. While conventional DTN simulators predominantly focus on simulating routing protocols, our approach integrates mobility modeling with DTN routing within the simulations. Connectivity among nodes is established based on various factors, including their geographical locations, communication ranges, and bit rates. The simulations encompass diverse categories of groups, such as vehicles and pedestrians, each characterized by a unique set of parameters. These parameters encompass attributes like message buffer size, radio range, and mobility model, which collectively contribute to the realism and accuracy of the simulated scenarios.

4.1. Simulation Environment

For the simulation of our proposed model, we have opted for the DakNet scenario, which replicates a network connecting three cities. Within these simulations, our attention is directed towards evaluating three key parameters: delivery probability, overhead ratio, and latency. To benchmark the efficacy of our proposed algorithm, we have conducted comparative analyses against several prominent DTN routing algorithms, all implemented within the ONE simulator. These algorithms include Spray and Wait, Prophet, Epidemic, and Fresh [2,8,27]. In this simulation setup, the Ping Pong application plays a pivotal role. It offers configurable options, allowing for the transmission of pings at fixed intervals or solely responding to received pings. When the application receives a ping, it promptly sends a corresponding pong message in return, thereby facilitating communication and interaction within the simulated network environment. The scenario simulates three rural cities lacking communication infrastructure, relying solely on opportunistic networking. Nodes communicate using Bluetooth interface at a data rate of 250 KB/sec with a radio range of 10 meters.

Group Configuration:

Group 1, 2, and 3: Pedestrians with random speeds between 0.5-1.5 m/s and pause times between 0 and 120 seconds. Each group corresponds to one city, with 40 hosts per city.

Group 4: Trams moving at speeds of 3-5 m/s with pause times between 10 and 30 seconds. Group 4 consists of 5 nodes.

Groups 1, 2, and 3 have up to 5 MB of free buffer space, while Group 4 nodes have 50 MB of free space for relaying messages.

Message Generation:

A new message is generated on average every 25 to 35 seconds.

Message size varies between 50 KB and 150 KB, with a time to live set to 300 minutes.

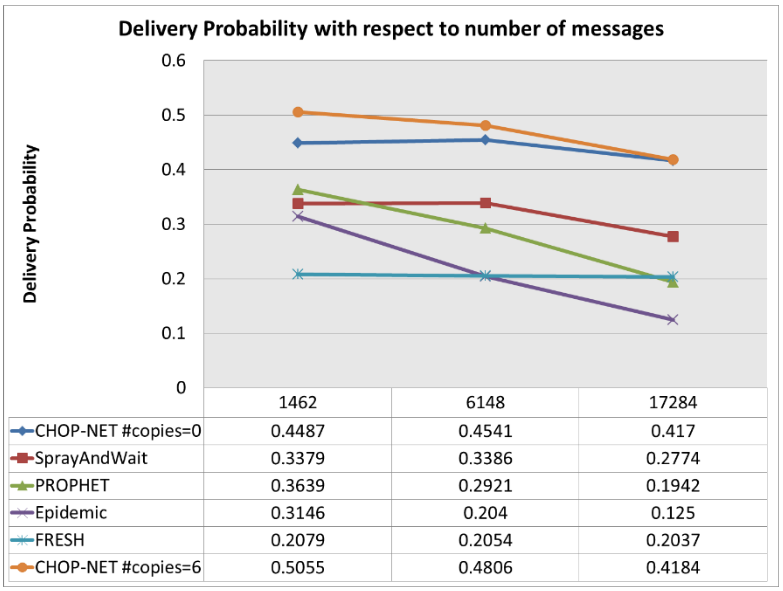
Simulation Duration:

Each simulation runs for 43200 seconds (12 hours).

4.2. Simulation Results

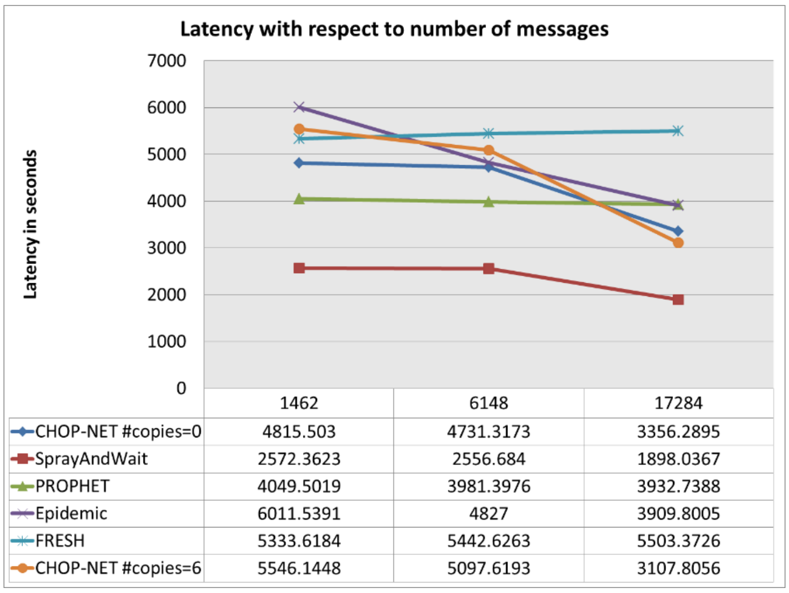
A. Varying the number of messages: increase the number of messages from 1462→6146→17191.

Figure 4 shows that the proposed algorithm called CHOPNET (Controlled and History based Opportunistic NETwork) has the highest delivery probability compared to other algorithms, and it maintains the best delivery probability among other algorithms while the number of messages increases. The delivery probability is significant with respect to other algorithms. Figure 5 shows that the proposed algorithm has an acceptable latency compared to other algorithms and it has a low and challenging overhead ratio as shown in Figure 6.



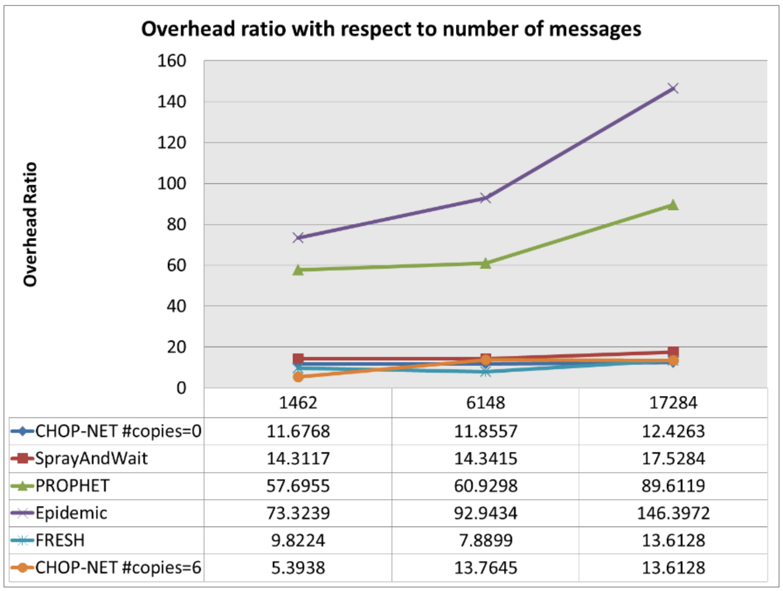
	1462	6148	17284
CHOP-NET #copies=0	0.4487	0.4541	0.417
SprayAndWait	0.3379	0.3386	0.2774
PROPHET	0.3639	0.2921	0.1942
Epidemic	0.3146	0.204	0.125
FRESH	0.2079	0.2054	0.2037
CHOP-NET #copies=6	0.5055	0.4806	0.4184

Figure 4. Delivery probability with respect to number of messages.



	1462	6148	17284
CHOP-NET #copies=0	4815.503	4731.3173	3356.2895
SprayAndWait	2572.3623	2556.684	1898.0367
PROPHET	4049.5019	3981.3976	3932.7388
Epidemic	6011.5391	4827	3909.8005
FRESH	5333.6184	5442.6263	5503.3726
CHOP-NET #copies=6	5546.1448	5097.6193	3107.8056

Figure 5. Latency with respect to number of messages.



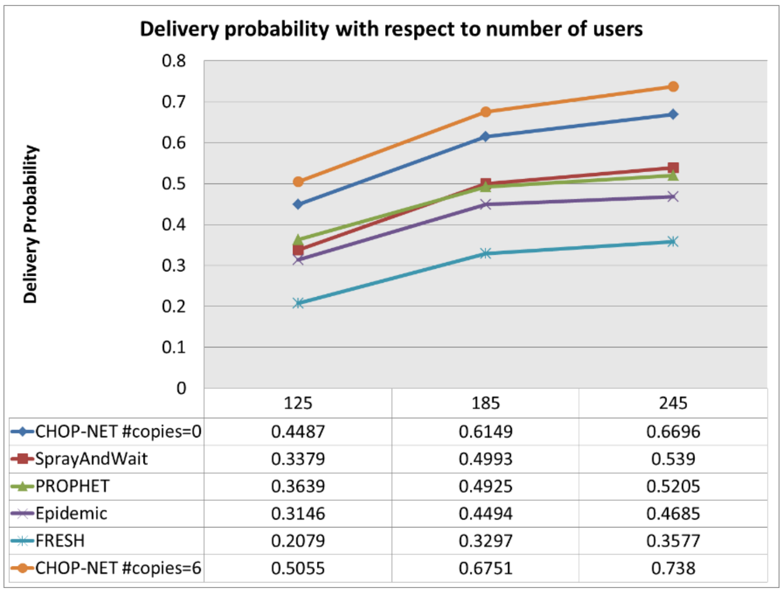
	1462	6148	17284
CHOP-NET #copies=0	11.6768	11.8557	12.4263
SprayAndWait	14.3117	14.3415	17.5284
PROPHET	57.6955	60.9298	89.6119
Epidemic	73.3239	92.9434	146.3972
FRESH	9.8224	7.8899	13.6128
CHOP-NET #copies=6	5.3938	13.7645	13.6128

Figure 6. Overhead ratio with respect to number of messages.

B. Varying the number of users:

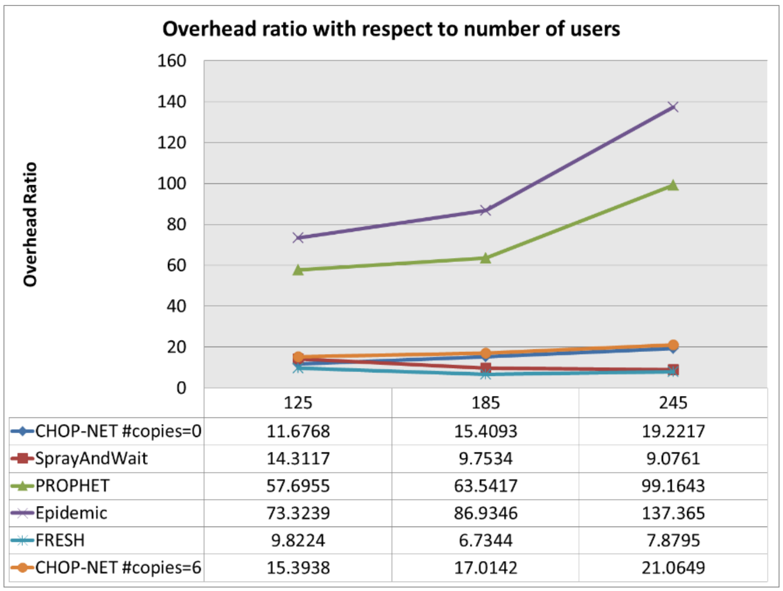
As illustrated in Figure 7, the proposed algorithm maintains the highest delivery probability as the number of users increases. For 245 users, the proposed algorithm with number of copies set to 0 has 0.66 delivery probability. It is shown that increasing the number of copies to 6 increases the delivery probability to 0.738. Figure 8 shows that the overhead ratio is low and challenging with respect to other algorithms. Figure 9 shows that the latency is acceptable with respect to other algorithms.

Note: Since some of the nodes travel between the cities, varying the transmission speed and the buffer size of all the nodes does not lead to a remarkable changes in delivery probability, overhead and latency; this is why these simulations are not discussed here.



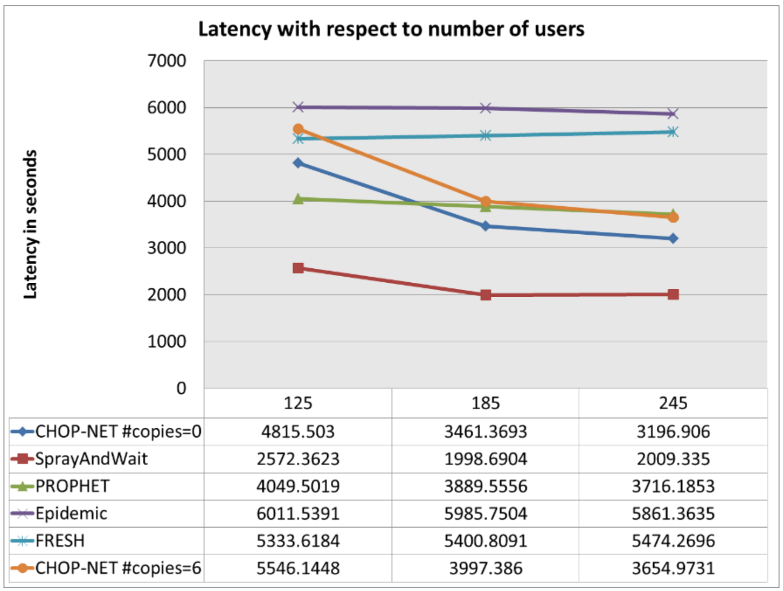
	125	185	245
CHOP-NET #copies=0	0.4487	0.6149	0.6696
SprayAndWait	0.3379	0.4993	0.539
PROPHET	0.3639	0.4925	0.5205
Epidemic	0.3146	0.4494	0.4685
FRESH	0.2079	0.3297	0.3577
CHOP-NET #copies=6	0.5055	0.6751	0.738

Figure 7. Delivery probability with respect to number of users.



	125	185	245
CHOP-NET #copies=0	11.6768	15.4093	19.2217
SprayAndWait	14.3117	9.7534	9.0761
PROPHET	57.6955	63.5417	99.1643
Epidemic	73.3239	86.9346	137.365
FRESH	9.8224	6.7344	7.8795
CHOP-NET #copies=6	15.3938	17.0142	21.0649

Figure 8. Overhead ratio with respect to number of users.

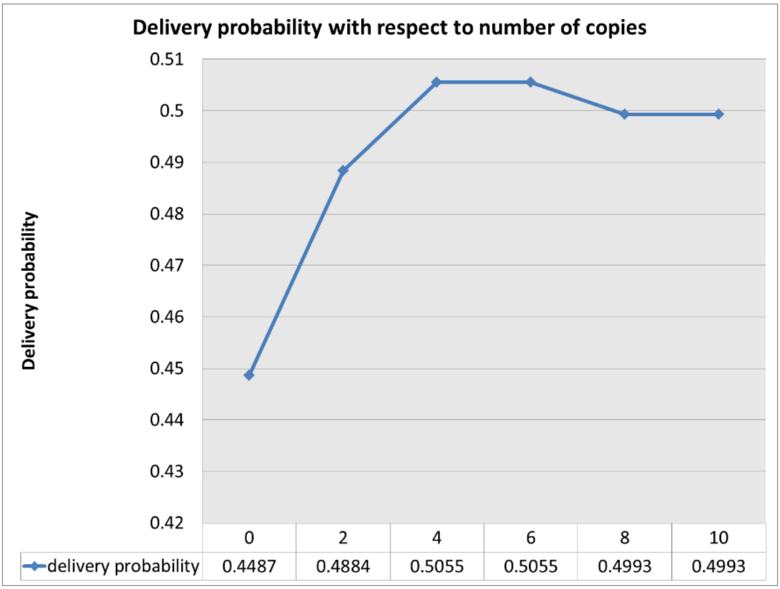


	125	185	245
CHOP-NET #copies=0	4815.503	3461.3693	3196.906
SprayAndWait	2572.3623	1998.6904	2009.335
PROPHET	4049.5019	3889.5556	3716.1853
Epidemic	6011.5391	5985.7504	5861.3635
FRESH	5333.6184	5400.8091	5474.2696
CHOP-NET #copies=6	5546.1448	3997.386	3654.9731

Figure 9. Latency with respect to number of users.

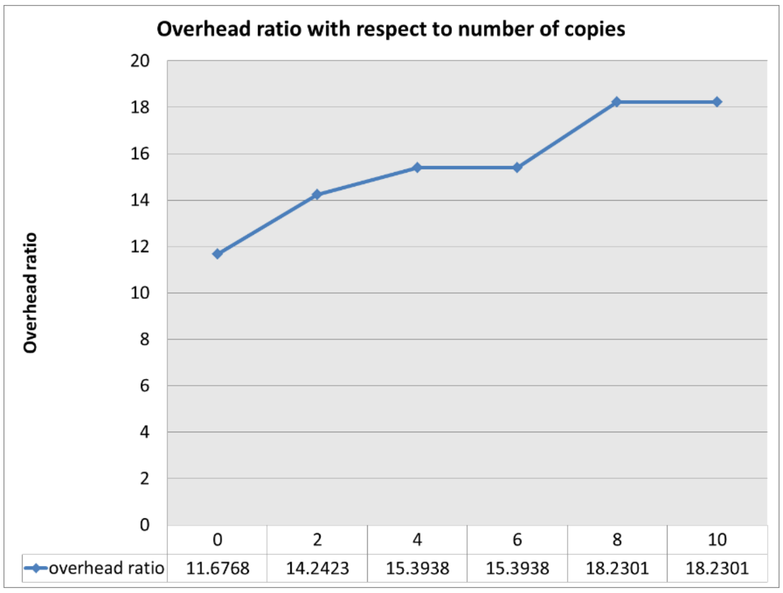
C. Varying the number of copies of the messages:

Figure 10 shows that increasing the number of copies of messages between 4 and 6 give the highest delivery probability. Increasing the number of copies beyond 6 increases the messages processing and transmission time and thus decreases the delivery probability. Figure 11 shows that increasing the number of messages increases the overhead; this is of because more messages are sent to deliver a single message. Also the latency increases as the number of copies increases as shown in Figure 12.



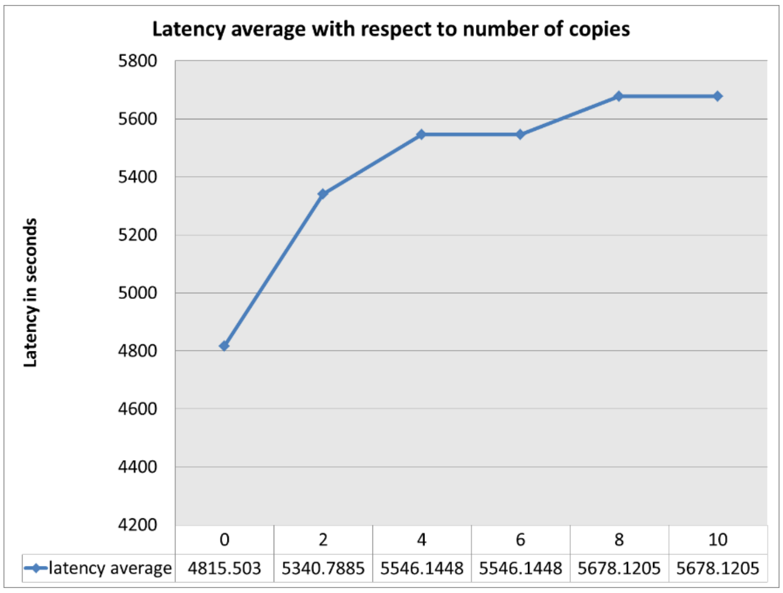
	0	2	4	6	8	10
delivery probability	0.4487	0.4884	0.5055	0.5055	0.4993	0.4993

Figure 10. Delivery probability with respect to number of copies (DakNet scenario).



	0	2	4	6	8	10
overhead ratio	11.6768	14.2423	15.3938	15.3938	18.2301	18.2301

Figure 11. Overhead ratio with respect to number of copies.,.



	0	2	4	6	8	10
latency average	4815.503	5340.7885	5546.1448	5546.1448	5678.1205	5678.1205

Figure 12. Latency with respect to number of copies.

D. Wifi-Direct scenario: Transmission speed 5 MB/sec, transmission range 30m, and buffer size 20 MB.

In this scenario also the proposed algorithm maintains the highest delivery probability (between 0.94 and 0.93) among the other algorithms. Notice that Spray and Wait has the lowest delivery probability (Figure 13); this is due to the randomness in spreading the messages in the network, while this scenario depends on smartly spreading the messages to the right nodes. Figures 14 and 15 show that the proposed algorithm achieves low overhead and latency compared to other algorithms.

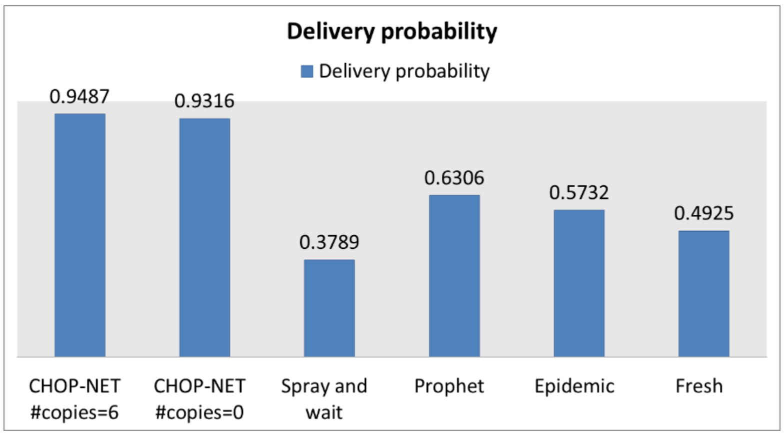


Figure 13. Delivery probability case wifi-direct.

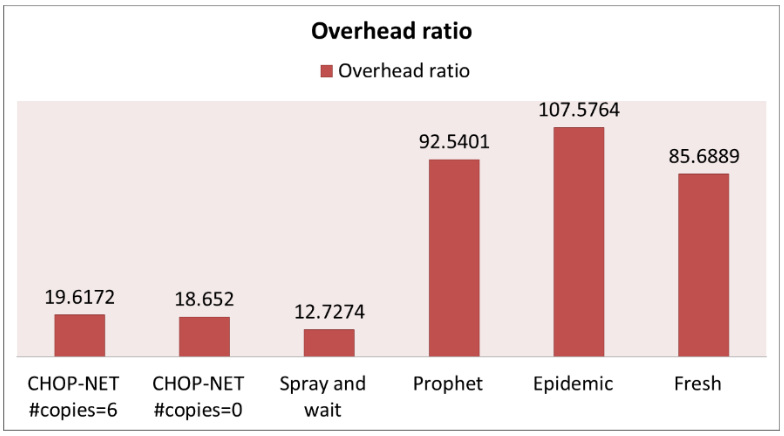


Figure 14. Overhead ratio case wifi-direct.

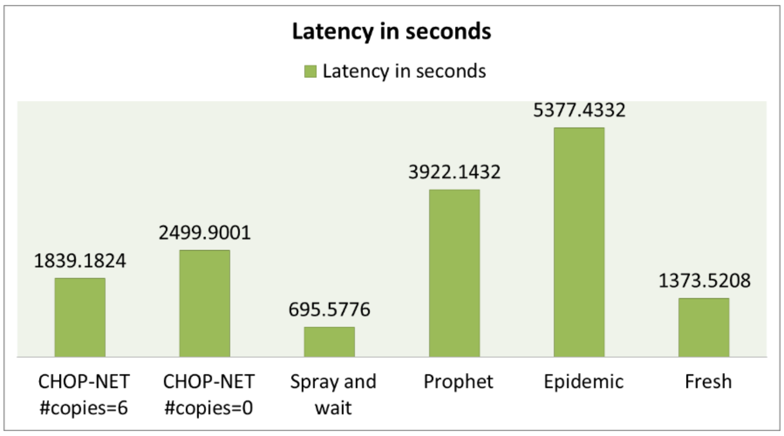


Figure 15. Latency case wifi-direct.

5. Conclusion

In this paper, we have introduced a novel probabilistic routing model for opportunistic networks. The new solution leverages nodes' meeting probabilities to dynamically route packets towards their destinations, effectively addressing the unique challenges posed by the absence of complete paths and rapidly changing connectivity in opportunistic networks. By incorporating historical data, specifically the meeting times between nodes, the model establishes controlled and efficient routing strategies, aiming to connect disjointed nodes within the network. Additionally, the proposed model integrates cross-layer optimization techniques, combining optimization strategies spanning multiple protocol layers. This integration aims to enhance energy efficiency, adaptability, and the reliability of message delivery within the network. Moreover, an efficient message detection scheme is introduced to alleviate high overhead by selectively deleting messages from buffers during congestion. Extensive simulations, have demonstrated the effectiveness of our proposed solution. Our results indicate that the new model achieves the highest delivery probability among prominent DTN routing protocols while maintaining acceptable and controlled overhead. As a result, the latency incurred remains within acceptable bounds, primarily attributed to the negotiation and table exchange processes between connected nodes. As a conclusion, the proposed model's applicability extends to IoT applications deployed in challenging environments, contributing to enhancing connectivity and efficiency in such deployments.

References

1. J. Gubbi et al., "Internet of Things (IoT): A vision, architectural elements, and future directions," in Future Generation Computer Systems, vol. 29, no. 7, pp. 1645-1660, 2013.
2. A. Mtibaa et al., "Opportunistic Networks: A Survey," in IEEE Communications Surveys & Tutorials, vol. 21, no. 3, pp. 2333-2362, 2019.
3. A. Mtibaa et al., "IoT-Enabled Opportunistic Networks: Applications and Challenges," in IEEE Internet of Things Journal, vol. 8, no. 2, pp. 907-924, 2021.
4. S. Pal and S. Sharma, "IoT-Based Disaster Management: A Survey," in IEEE Transactions on Sustainable Computing, vol. 6, no. 2, pp. 206-219, 2021.
5. S. Chatterjea et al., "Wildlife Monitoring Using Opportunistic Networks: A Review," in IEEE Access, vol. 8, pp. 106091-106108, 2020.
6. M. Conti et al., "Rural Connectivity: Challenges and Opportunities," in IEEE Communications Magazine, vol. 59, no. 4, pp. 120-126, 2021.
7. C. Boldrini et al., "Connectivity Prediction in Opportunistic Networks: A Survey," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1148-1171, 2020.
8. Y. Sun et al., "Dynamic Routing Protocols for Opportunistic Networks: A Review," in IEEE Transactions on Mobile Computing, vol. 20, no. 4, pp. 1404-1419, 2021.
9. A. Boukerche et al., "Energy-Efficient Routing Protocols for Opportunistic Networks: A Survey," in IEEE Transactions on Mobile Computing, vol. 20, no. 6, pp. 1847-1865, 2021.
10. S. Agarwal et al., "Content-Centric Routing in IoT-Enabled Opportunistic Networks," in IEEE Transactions on Vehicular Technology, vol. 71, no. 2, pp. 1146-1159, 2023.
11. M. D. Zennaro et al., "Cross-Layer Optimization in IoT-Enabled Opportunistic Networks," in IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 1341-1355, 2021.
12. M. Shojafar et al., "Edge Computing Integration in IoT-Enabled Opportunistic Networks: Challenges and Opportunities," in IEEE Transactions on Cloud Computing, vol. 10, no. 4, pp. 1766-1780, 2022.
13. X. Zhang et al., "Context-Aware Routing Protocol for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 20, no. 5, pp. 1440-1453, 2021.
14. Y. Li et al., "Hybrid Routing Protocol with Social-Based Forwarding for Opportunistic Networks," in IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 9656-9668, 2021.
15. Z. Chen et al., "Mobility-Aware Routing Protocol for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 19, no. 8, pp. 1945-1957, 2020.
16. H. Wang et al., "Ant Colony Optimization Inspired Routing Protocol for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 18, no. 12, pp. 2967-2980, 2019.
17. Q. Yang et al., "Reinforcement Learning-Based Routing Protocol for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 20, no. 7, pp. 2509-2522, 2021.
18. Z. Liu et al., "Policy Routing Opportunistic Networks: A Deep Reinforcement Learning Approach," in IEEE Transactions on Vehicular Technology, vol. 70, no. 9, pp. 8885-8897, 2021.
19. Y. Zhao et al., "Energy-Aware Routing Protocol for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 18, no. 5, pp. 1096-1109, 2019.
20. J. Li et al., "Energy-Efficient Data Dissemination in Opportunistic Networks Based on Social Network Analysis," in IEEE Transactions on Vehicular Technology, vol. 68, no. 8, pp. 7765-7778, 2019.
21. H. Kim et al., "Energy-Efficient IoT Architecture with Edge Computing and Data Aggregation," in IEEE Internet of Things Journal, vol. 6, no. 4, pp. 6405-6414, 2019.
22. X. Zhang et al., "An Energy-Efficient MAC Protocol for IoT Networks," in IEEE Transactions on Industrial Informatics, vol. 15, no. 5, pp. 3080-3090, 2019.
23. Q. Liu et al., "Cross-Layer Optimization for Energy Efficiency in IoT Networks," in IEEE Internet of Things Journal, vol. 7, no. 3, pp. 1844-1854, 2020.
24. H. Wang et al., "Cross-Layer Energy Management for Opportunistic Networks," in IEEE Transactions on Mobile Computing, vol. 19, no. 7, pp. 1663-1675, 2020.

25. M. M. H. Morsi, S. Yasir and H. Hassanein, "ONE: An Integrated DTN and Mobility Simulation Framework," 2013 IEEE 78th Vehicular Technology Conference (VTC Fall), Las Vegas, NV, USA, 2013, pp. 1-5, doi: 10.1109/VTCFall.2013.6692039.
26. G. Parissidis, M. A. Zuniga and T. Spyropoulos, "Inter-connection of DTN and Infrastructure Networks in ONE: Opportunistic Network Environment," 2010 IEEE Symposium on Computers and Communications (ISCC), Riccione, 2010, pp. 759-764, doi: 10.1109/ISCC.2010.5546712.
27. A. Khalil and B. Zeddini, "A Secure Opportunistic Network with Efficient Routing for Enhanced Efficiency and Sustainability," Future Internet, vol. 16, p. 56, 2024.