TECHNISCHE
UNIVERSITÄT
DARMSTADT

SCALABLE AND SECURE MULTICAST ROUTING FOR MOBILE AD-HOC NETWORKS

MILAN SCHMITTNER

Master Thesis

30. September 2014

Secure Mobile Networking Lab
Department of Computer Science

Scalable and Secure Multicast Routing for Mobile Ad-hoc Networks
Master Thesis
SEEMOO-MSC-0049

Submitted by Milan Schmittner
Date of submission: September 30, 2014

Advisor: Prof. Dr.-Ing. Matthias Hollick
Supervisor: Dipl.-Ing. Michael Noisternig

Technische Universität Darmstadt
Department of Computer Science
Secure Mobile Networking Lab

ABSTRACT

Mobile Ad-Hoc Networks (MANETs) are decentralized and autonomous communication systems: They can be used to provide connectivity when a natural disaster has brought down the infrastructure, or they can support freedom of speech in countries with governmental Internet restrictions. MANET design requires careful attention to scalability and security due to low-capacity and error-prone wireless links as well as the openness of these systems.

In this thesis, we address the issue of multicast as a means to efficiently support the MANET application of group communication on the network layer. To this aim, we first survey the research literature on the current state of the art in MANET routing, and we identify a gap between scalability and security in multicast routing protocols—two aspects that were only considered in isolation until now. We then develop an explicit multicast protocol based on the design of a secure unicast protocol, aiming to maintain its security properties while introducing minimal overhead.

Our simulation results reveal that our protocol reduces bandwidth utilization in group communication scenarios by up to 45 % compared to the original unicast protocol, while providing significantly better resilience under blackhole attacks. A comparison with pure flooding allows us to identify a practical group size limit, and we present ideas for better large-group support.

ZUSAMMENFASSUNG

Mobile Ad-hoc Netzwerke (MANETs) sind dezentrale und autonome Kommunikationssysteme. Sie können z. B. genutzt werden, um die durch Naturkatastrophen zerstörte Infrastruktur für sowohl Ersthelfer als auch die Bevölkerung (temporär) zu ersetzen. Sie ermöglichen außerdem die Umgehung von Internetzensur im Sinne der Meinungsfreiheit. Bei der technischen Umsetzung solcher Systeme ist es wichtig, die Eigenheiten im Bezug auf Skalierbarkeit und Sicherheit zu verstehen und zu beachten, welche hauptsächlich durch die Fehleranfälligkeit und die geringe Kapazität drahtloser Kommunikationskanäle und die prinzipielle Offenheit von MANETs gegeben sind.

Diese Masterarbeit soll das Problem der effizienten Gruppenkommunikation mit Hilfe von Multicast auf der Vermittlungsschicht lösen. Dazu wurden zunächst aktuelle Forschungsergebnisse im Bereich MANET-Routing ausgewertet und dabei erkannt, dass noch keine Arbeiten zu gleichzeitig skalierendem *und* sicherem Multicast veröffentlicht wurden. Das in dieser Arbeit entwickelte explizite Multicast-Protokoll basiert auf einem sicheren Unicast-Protokoll, wobei versucht wurde, den zusätzlichen Kommunikationsaufwand möglichst gering zu halten.

Die Simulationsergebnisse zeigen, dass die Multicast-Lösung, verglichen mit dem ursprünglichen Unicast-Protokoll, den Bandbreitenbedarf um bis zu 45 % reduziert und gleichzeitig eine höhere Resistenz gegen Blackhole-Angriffe aufweist. Außerdem wurde erkannt, dass expliziter Multicast ab einer gewissen Gruppengröße keinen Effizienzvorteil gegenüber einem einfachen Flooding-Protokoll aufweisen kann. Anhand dessen wurden Lösungen vorgeschlagen, um die maximal unterstützte Gruppengröße weiter zu erhöhen.

ERKLÄRUNG ZUR MASTER-THESIS

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, 30. September 2014

Milan Schmittner

CONTENTS

1 INTRODUCTION

Mobile Ad-Hoc Networks (MANETs) are a well-studied research field. Most work has been carried out in trying to adapt classic routing paradigms towards operating more efficiently in hostile, infrastructure-less, and resource-constrained environments. Scalability issues such as the flooding problem have been identified and addressed in various ways [96, 40, 95].

Whereas classic IP routers are usually operated by some trusted third party, e.g., an Internet provider, security is of increased concern in MANETs where other (potentially untrusted) parties take on the role of a router. An adversary that propagates false routing information could single-handedly bring down such a network. As a result, solutions towards secure routing have been proposed starting from around 2002. Often, such solutions impose impractical security assumptions on the system or harden the network only against a specific kind of attack. Approaches that follow the security-by-design principle can achieve a more comprehensive attack resistance without becoming impractical, e. g., [31].

1.1 MOTIVATION

Apart from the military domain, civil projects have recently deployed infrastructure-less networks: *Freifunk* in Germany [28] is a movement towards creating an alternative communication network, which shall be anonymous, free to use for everyone, and free from discrimination (net neutrality). Freifunk routers are maintained by volunteers and run MANET routing protocols to achieve interconnectivity. *FireChat* [75], as a second example, is a MANET-based group chat application for mobile devices. It can be used as a tool for emergency communication or even to circumvent governmental Internet restrictions [7]. However, it is not secure [17].

Multicast is a mechanism that efficiently enables group communication: A sender can address a single message to multiple destinations without the need for per-destination transmissions (= unicast), reducing bandwidth utilization significantly. However, multicast poses its own challenges: How to find routes to all destinations? How is group membership maintained? In particular, how can changes be handled efficiently? How to make multicast robust against MANET deficiencies? How can such a protocol be secured against adversaries?

1.2 CONTRIBUTION

In this thesis, we examine the state of the art of MANET routing in terms of *scalability* and *security* in an orthogonal manner. We give a comprehensive overview and a discussion of scalability improvement mechanisms for both unicast and multicast routing protocols. The various proposals are categorized according to their core ideas. We investigate which concepts are promising and which could even be combined with others for the construction of more sophisticated solutions. Security-related (unicast and multicast) protocols are compared according to the protocol authors' security assumptions. We also review the resistance of the various protocols against common attacks. We conclude that secure MANET multicast routing is an understudied research area.

Following these findings, we design a scalable *and* secure multicast protocol based on Castor [31], a promising unicast routing protocol for MANETs. We implement both Castor and our multicast-enabled protocol *Xcastor* in the Click modular router framework. We evaluate the performance of our protocol by comparing it to Castor and a flooding protocol in ns-3.

1.3 OUTLINE

In Part I of this thesis (Chapters 2 and 3), we try to answer the following questions: What is scalability? What problems do MANETs impose in terms of scalability? How can we approach and eventually solve these issues? How is security affected by these solutions?

The first two chapters of Part II (Chapters 4 and 5) present the design process and the implementation of our protocol. We evaluate and discuss our results in Chapters 6 and 7. The thesis is concluded in Chapter 8.

Part I

STATE OF THE ART: SCALABILITY AND SECURITY OF MANET ROUTING

2 SCALABILITY OF MANET ROUTING

The main advantage of Mobile Ad-Hoc Networks (MANETs) is their capability of rapid deployment and operation without a fixed infrastructure. It allows these networks to be deployed in scenarios, where other means of communication are not feasible or simply not available.

However, such networks are inherently poorly scalable. This has several reasons: For one, the wireless channel is a shared medium. Therefore, participants mutually exclude each other from concurrent transmission. Generally, wireless technology exhibits less bandwidth and is less reliable due to channel errors than its wired counterpart. The participating devices (*nodes*) are usually battery-powered, which means that they might unexpectedly fail. Last but not least, node mobility causes frequent topology changes which needs to be mitigated[1].

This chapter is structured as follows: In Sections 2.1 and 2.2, we introduce the notion of scalability and describe the system model considered. In Section 2.3, we review general limitations of scalability in MANETs and based on which we analyze existing approaches towards increasing scalability for unicast (Section 2.4) and multicast (Section 2.5) routing protocols.

2.1 DEFINITION

We refer to *scalability* as the ability of a network to grow with the number of participating nodes n while remaining operational and efficient. In particular, efficiency corresponds to a low consumption of available bandwidth for non-user data transfer, and low processing and memory requirements per node. In other words, in a scalable network, the per-node bandwidth as well as processing and memory usage are not severely affected by n. In a poorly scalable network, nodes either suffer from low available bandwidth, high processing demands, full memory, or a combination of these as n grows.

Within this chapter, we consider the conservation of available bandwidth as the primary goal of a scalable routing protocol. The consumption of computing and memory resources receives minor consideration but will be looked at in more detail in Chapter 3.

2.2 SYSTEM MODEL

For our system model that we consider in this thesis, we assume a MANET with the following properties:

1. A *shared communication channel*. All nodes are using the same technology for wireless transmissions. This requires the presence of an access control mechanism at the data link layer.

1 We will later see that scalability can actually benefit from mobility.

2. *Broadcast communication*. All nodes are equipped with omnidirectional antennæ so that a transmission can be overheard by any neighbor. The transmission *range* does not vary, i. e., links are bidirectional.

3. Nodes *can* have different amounts of available resources in terms of battery, memory or processing power. This means that some nodes can have more capacities than others and, thus, might be suitable to perform special tasks.

4. *Continuously connected network*. This means that there always exists a (multihop) path between any two nodes in the network. This is in contrast to the notion of Delay Tolerant Networks (DTNs), where network partitioning is expected and the existence of end-to-end paths is the exception.

2.3 LIMITS OF SCALABILITY

In the introduction of this chapter, we have stated a number of limitations of MANETs in terms of scalability. Below, we provide theoretical background explaining these limitations—and also argue for approaches to circumvent them on the network layer.

2.3.1 *Network Layer Scalability Limits Imposed by Lower Layers*

Gupta and Kumar have shown that a capacity boundary for routing in wireless ad hoc networks exists [38]. This boundary derives from broadcast communication and the resulting interference of concurrently transmitting nodes. The wireless channel is a shared medium, so local concurrent transmissions cause collisions, or—if a medium access control scheme is deployed—nodes block each other from concurrent transmissions. Gupta and Kumar have shown that even in the case of optimal node placement, traffic patterns and scheduling, the global network capacity grows as $\Theta(W\sqrt{n})$, where W is the bandwidth of a single node and n the number of nodes in the network. This means that the average throughput for a single node reduces to the order of $\Theta(W/\sqrt{n})$. For random networks, the per-node capacity even reduces to $\Theta(W/\sqrt{n \log n})$: Even under optimal circumstances, the per-node capacity approaches zero as the network grows.

From their findings, the authors see two solutions towards circumventing the capacity problem: *1*) to deploy networks with only a small number of nodes *or* *2*) to deploy networks that exhibit traffic locality; i. e., nodes may often communicate with other nodes that are geographically close and transmissions to distant parts of the network are scarce. This leads to less interference because intermediate nodes are not required to relay messages so often.

In their model, Gupta and Kumar assume nodes to be static and do not allow topology changes, which disregards node mobility altogether. However, mobility is an inherent property of MANETs. Grossglauser and Tse show that mobility can in fact increase the scalability of MANETs dramatically [36]. The rationale is the following: When nodes move around, then, at some point in time, a source-destination pair will be close enough to perform direct communication. This is when data can be exchanged—avoiding the use of relays. Grossglauser and Tse show that using no relays at all is not sufficient in order to achieve maximum throughput because the probability of such meetings is too low. However, a single (and thus constant) level of relays, i. e., two-hop communication, is sufficient to achieve a constant per-node throughput of $O(W)$. The relays increase the

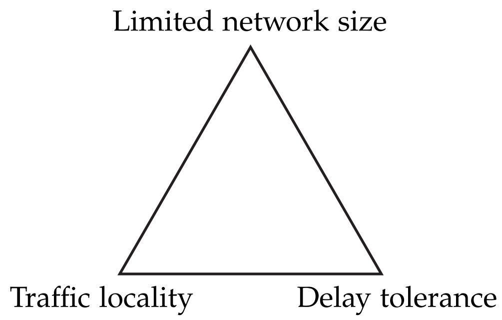


Figure 2.1: Dimensions allowing for optimizing MANET scalability in the model by Gupta and Kumar, Grossglauser and Tse.

probability of the source message being delivered to the destination. Note that the delay introduced by this forwarding scheme is unbounded and increases with the network size. Thus, the constant throughput achieved by mobility is a long-term average. Consequently, trading off delay for throughput gains is only feasible for network applications that can handle significant amounts of delay, e. g., E-mail.

From a theoretical point of view, we have now reviewed three factors that can independently affect the capacity of MANETs: *1*) The network size, *2*) locality of the traffic, and *3*) delay tolerance (see Figure 2.1).

Let us investigate these factors more closely. The network size is usually dependent on the application, e. g., whether the network is deployed in a large stadium with ten thousands of nodes or in a small classroom. The network size is, in any case, not controllable by the routing protocol. The locality of traffic is also dependent on the application, i. e., the generated traffic patterns [108] but it is also affected by the communication overhead of the routing protocol. Thus, the locality of dissemination of routing information has an impact on scalability, which can be dealt with. Whether network delay is acceptable is equally application-dependent. This property can be exploited by routing protocols to reduce the overall load on the network.

Since the discussion of DTNs is out of scope of this thesis due to the complexity of this topic alone, the only optimization factor left is traffic locality. As a result, all further discussion in this chapter is based on the principle of *keeping traffic local with respect to other transmissions by keeping path lengths short* and *minimizing the amount of overall data dissemination*.

2.3.2 *Circumventing the Scalability Curse*

The network layer limitations originate from the assumption that neighboring nodes cannot transmit concurrently. If we remove these assumptions, the scalability curse on the network layer vanishes. We mention two examples that allow concurrent transmissions.

Interference Alignment. Multiple (K) sender-receiver pairs can actually use the medium at the same time by using a linear precoding technique called interference alignment [12]. In theory, a maximum of $K/2$ interference free links are achievable. Work towards practical realization is currently carried out, e. g., [67].

MAC Capture Effect. The capture effect [103] is the observation that two concurrently transmitted packets with different signal strengths do not necessarily cause packet loss: If the stronger packet is being received first, the second (and superimposed) packet does not cause corruption. Sparkle [106], for example, exploits this effect to enable fast and reliable communication in Wireless Sensor Networks.

These techniques are still under research. Current MANETs make use of Media Access Control (MAC) mechanisms that strictly preclude concurrent transmissions, e. g., IEEE 802.11's distributed coordination function (DCF) [46]. Thus, we assume the model by Gupta and Kumar in this thesis.

2.4 SCALABILITY OF MANET UNICAST ROUTING

This section explores scalability improvement mechanisms for the classical routing principles: Distance Vector Routing (DVR), Link State Routing (LSR), and Source Routing (SR). Most of the work on MANET routing has been done in this area, and it is protocols based on these principles that have been published in the IETF[2] [87, 18, 54].

Surveys on MANET routing already exist, e. g., [41]. We argue that our more analytical approach is necessary to understand state of the art MANET routing in the light of scalability.

2.4.1 *Fundamentals*

Unicast routing is a communication technique that enables a node to communicate with another node in the network, without requiring the other node to be within direct transmission range, i. e., the two nodes need not to be *neighbors*. In this case, one or multiple other nodes are required to act as relays and to forward the message to the destination. A routing protocol is enacted to find such nodes, which comprise an ordered list of neighboring relay nodes, and which we call *route* or *path*.

2.4.1.1 *Routing Principles*

Below, we briefly describe the classical routing principles (Distance Vector Routing, Link State Routing and Source Routing) as well as more exotic ones.

Distance Vector Routing (DVR) is based on exchanging routing table entries (containing distance vectors) between neighbors in a network. This enables a node to construct a path to the destination by exploiting one of the neighbors' paths. The basic idea of finding routes in DVR is as follows: Let s and d be source and destination nodes, respectively. If a neighbor h of s knows and announces some path p to d, then s knows that it can reach d over h using the path $p' = (h, p)$.

DVR suffers from slow convergence (because routing state updates are propagated one hop at a time) and a tendency of creating routing loops when using hop count as a metric [83]. However, such protocols are more efficient with respect to message overhead compared to Link State Routing because no flooding is required.

The Destination-Sequenced Distance-Vector (DSDV) Routing [85] protocol is one of the first proposed solutions for MANET routing. Its main optimization compared to traditional DVR protocols is loop-freedom, achieved by introducing sequence

2 Experimental RFC

numbers. Another well-known protocol is Ad hoc On-Demand Distance Vector (AODV) routing [86, 87]. It is similar to DSDV but makes use of on-demand routing (which will be discussed later).

Link State Routing (LSR) is around since the 1980s [70] and has been successfully deployed in the Internet in the form of the Open Shortest Path First (OSPF) [71] and Intermediate System to Intermediate System (IS-IS) [22] protocols. LSR is based on a periodical exchange of neighbor connectivity information, which is flooded through the network in the form of Link State Update (LSU) packets. This enables every node to receive a global view of the network topology, which allows it to calculate shortest routes, e. g., using Dijkstra's algorithm [25].

The main problem of deploying LSR in a MANET environment is the flooding of link-state information: The communication overhead for an unmodified LSR protocol is in the order of $O(n^2)$, i. e., n (re)broadcasts for n nodes (with n being the number of nodes in the network).

The best-known example for MANET Link State Routing is the Optimized Link State Routing (OLSR) [51, 18] protocol. It uses the concept of *Multipoint Relays* for improving efficiency, which is explained in Section 2.4.2.

Source Routing. An alternative approach to DVR and LSR is so-called Source Routing (SR). In SR, the source of a packet specifies the route the packet has to take to its destination. This is in contrast to DVR and LSR where routing decisions are made at the intermediate routers. With this control, a source is able to route around, e. g., "dangerous" territory, thus, enforcing certain route policies.

The best-known example in this category is the *Dynamic Source Routing (DSR)* protocol [53, 54].

Other. The open-source implementation of Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) [73] enables routing for low-power embedded devices by removing the need for expensive recalculation of the topology map as in OLSR [76]. B.A.T.M.A.N. nodes periodically announce their presence by broadcasting lightweight Originator Messages (OGMs) to their neighbors. Each OGM receiver locally decides whether to forward the packet, based on whether the sender is considered the best next hop to the OGM originator w. r. t. link quality[3]. This selective forwarding results in full dispersion of OGMs in the network without the need for expensive flooding. The resulting routing tables at each node contain next hop information to every other node in the network. Thus, B.A.T.M.A.N. can be seen as a relative to DVR.

Researchers have also considered routing based on *ant colony optimization*. The nature inspired approach follows the observation of how ants find a shortest path from their nest to a food source. An example ant colony optimization-based protocol is AntHocNet [13].

3 Note the difference between *sender* (any intermediate node forwarding an OGM) and *originator* (the source of an OGM).

2.4.1.2 *Route Discovery Initiation*

We describe the two principle ways of how route discovery can be initiated: *proactively* and *reactively*.

Proactive. Traditional Internet routing protocols are *proactive* or *table-driven* protocols. They are based on a periodical exchange of routing information from which each router in the network can gather a broad view of the network topology and thus can have routing entries to every destination readily available.

Examples for this type are DSDV and OLSR.

Reactive. Another class of MANET routing protocols have adopted the concept of *reactive* or *on-demand* route discovery. The idea is based on the fact that keeping routing tables up-to-date when they are not needed is inefficient. In addition, it is unnecessary to have routes to *all* other nodes in the network because communication typically takes place only with a subset of nodes. Reactive protocols discover routes when they are needed.

Two well-known protocols in this class are AODV and DSR. Both protocols rely on a similar route discovery scheme that is based on Route Request (RREQ) and Route Reply (RREP) packets. If no routing table entry exists to a specific destination prior to sending out a data packet, a RREQ packet is flooded in the network specifying the sought destination. In AODV, forwarding nodes will store the *reverse* path in their routing tables, i. e., the previous hop of the packet. In DSR, on the other hand, the list of intermediate nodes is stored and forwarded in the RREQ. In both cases, if the RREQ reaches its destination, the destination generates a RREP packet that is sent along the reverse path. In AODV, the RREP updates the routing tables of intermediate nodes which then have a valid entry for the *forward* path to the destination. In DSR, the source route is contained in the RREP which is routed back over the reverse path.

Proactive routing maintains up-to-date state and is preferable if *1*) many nodes actively communicate and *2*) nodes frequently move in the network.

Reactive protocols, on the other hand, find routes only when needed. Consequently, there is no overhead until data transmission takes place. However, they introduce a delay because the route discovery process has to be run first[4]. Reactive routing should thus be deployed if only few paths are in use.

Some protocols take a *hybrid* approach by combining the two strategies, e. g., by using a proactive strategy in the close-by environment and relying on reactive routing for remote destinations.

2.4.2 *Improving Scalability*

All of the above mentioned MANET routing protocols rely on some sort of flooding mechanism. Proactive protocols use it for dissemination of routing information throughout the network while on-demand protocols use flooding to find the destination in the route discovery phase. Efficient packet dissemination is, hence, most crucial for network performance as one message impacts the entire network.

4 This can partly be mitigated if *piggybacking* is used, i. e., user data is appended to the RREQ packets.

As discussed in Section 2.3.1, traffic has to remain local and sparse for a routing protocol to be scalable. In the literature, there are two principle concepts aiming towards this goal. All of the proposals optimize flooding in either or both of the following dimensions:

Spatial limitation. Most proposals employ mechanisms that limit the number of nodes involved in the flooding phase. This can be done either by limiting the flooding to a certain *geographical* area, e. g., a circle around the source, or by *selecting* appropriate forwarding nodes.

Temporal limitation. Some proposals reduce flooding overhead by lowering the frequency at which packets are transmitted. On-demand routing, for example, can already be seen as an optimization in this dimension. It avoids periodic updates and floods the network only when a route request is emitted.

In the following subsections, we present various approaches and exemplary protocols for implementing these concepts. We then discuss these approaches and compare them w. r. t. applicability and compatibility.

2.4.2.1 *Route Caching*

On-demand protocols rely on a route discovery process that is initiated before user data can be transmitted. This route discovery is costly for two reasons: *1*) It introduces a delay due to the preceding RREQ and RREP messages and, more severely, *2*) requires flooding of the entire network.

Caching of routing information at intermediate nodes can help to both speed up the route discovery process and to *spatially* limit the scope of flooding: When receiving a RREQ, any intermediate node with a fresh, that is, up-to-date route to the requested destination may immediately reply with a RREP to the source, thus avoiding further dissemination of the RREQ and speeding up the route discovery process. This mechanism is also known as *Gratuitous Route Replies* (an optimization proposed for AODV [8]).

Cache freshness is an issue, especially in volatile environments: Caches might be out-dated, thus providing misleading routing information to querying nodes.

2.4.2.2 *Expanding Ring Search*

Expanding Ring Search is an optimization proposed for AODV [8] but is generally deployable for any type of on-demand routing. The idea is to *spatially* limit the reach of RREQs, for example, by using a Time To Live (TTL) hop counter. Upon route discovery, a RREQ with a small TTL of 1 or 2 may be sent out. If the route discovery is unsuccessful (because the destination is farther away than TTL hops), another RREQ with an increased TTL is broadcast. This incremental process may continue until a certain TTL threshold is reached and the protocol falls back to full flooding. Instead of TTL, one could also use a distance boundary based on geographical coordinates.

Expanding Ring Search can increase scalability if the destinations are close-by most of the time so that the destination is found in an early iteration. If this is not the case, i. e., if it falls back to flooding most of the time, then the mechanism can actually perform worse than flooding because of the previous unsuccessful iterations. Furthermore, the mechanism adds additional delay to the search, growing with the number of iterations.

2.4.2.3 *Myopic Dissemination*

Some proactive protocols reduce the *frequency* at which routing updates are disseminated to *distant* nodes. This limits the dissemination in a *temporal* and *spatial* manner. The literature refers to it as "myopic dissemination". Protocols incorporating this mechanism have the drawback of relying on possibly imprecise routing information about distant nodes. However, this is not a severe problem as route information becomes more precise as a packet is approaching its destination.

FSR Fisheye State Routing (FSR) [50, 83] exchanges link-state information only with direct neighbors. Similar to proactive DVR (such as DSDV), LSU packets in FSR contain accumulated link-state information of all node pairs in the network. To decrease LSU packet sizes, FSR less frequently includes link-state entires of more distant[5] nodes. In FSR, information propagation converges slowly, i. e., by one hop per update interval.

FSLS The Fuzzy Sighted Link State (FSLS) protocol [96] sends out LSUs to distant nodes at a lower frequency. The authors derived an optimal solution in terms of overhead, the Hazy Sight Link State algorithm: The hop count of the LSU packet is set to 2^i for $i = 1, 2, 3, \ldots$ after every 2^{i-1} time units.

DREAM Basagni et al. introduced the Distance Routing Effect Algorithm for Mobility (DREAM) [5], which features a dissemination technique similar to FSLS. The main difference is that DREAM relies on the Global Positioning System (GPS) and limits the spread of update packets in terms of geographical distances, i. e., only nodes that are closer than some distance r are allowed to re-broadcast packets. Also, the update packets contain the nodes' locations rather than link-state information. Connectivity information is only kept to 1-hop neighbors. DREAM is based on what the authors call the "distance effect": Distant nodes appear to move slower relatively to each other than close-by nodes. Consequently, update packets to distant nodes are sent out less frequently. Based on the "mobility rate", i. e., how fast a node is moving, a node can autonomously determine the overall frequency of update packets. The actual routing procedure is GPS-aided and explained in Section 2.4.2.8.

ZRP The Zone Routing Protocol (ZRP) [39] is a hybrid approach that uses proactive and on-demand protocols on two levels. Proactive routing is used within a predefined radial zone around the node, i. e., a predefined maximum number of hops h: Periodic flooding of routing updates is restricted to nodes that are at most h hops away. ZRP uses an on-demand routing protocol to enable communication with nodes outside the radial zone. Even though ZRP establishes a hierarchy, it is not classified as a *clustering* protocol: The hierarchy is created locally per-node and does not require inter-node communication.

2.4.2.4 *Clustering*

Clustering introduces a hierarchy between nodes, i. e., there are usually two or more groups (or levels) of nodes with different responsibilities. This is in contrast to other

5 outside a predefined scope

protocols that have a *flat* structure. We first give some examples of clustering-based protocols and then discuss their benefits and drawbacks.

CGSR The Clusterhead-Gateway Switch Routing (CGSR) protocol [15] is based on DSDV. The protocol clusters the network by assigning two special roles to the nodes: *clusterheads* and *gateways*. Clusterheads are responsible for routing all traffic to and from nodes in the cluster. They are decentrally elected based on the proposed *Least Clusterhead Change* algorithm, which favors stable clusterheads in order to avoid frequent re-elections. Gateway nodes are members of more than one cluster: They act as connectors between the clusterheads. As a result, packets are routed alternatingly between clusterheads and gateways, which places significant load on these nodes. The authors propose a clusterhead-oriented token scheme on the data link layer to give priority to the clusterheads.

CGSR uses two tables for routing: A cluster member table and a DV table listing all clusterheads. Routing is then performed by looking up the responsible clusterhead for the destination node.

LANMAR Landmark Ad Hoc Routing (LANMAR) [84] incorporates similar concepts. In LANMAR, clusters are formed according to nodes' *group mobility*, i.e., nodes that travel together form a group. One of the nodes in each group is elected as a *landmark*. Each landmark node represents its group and corresponds to an entry in a global distance vector routing table. Nodes within a group are reached via their corresponding landmark nodes. Within a group, a myopic link-state-based protocol (here: FSR) is used. Having only clusterheads (CGSR) or landmark nodes (LANMAR) in the DV entries has two positive side effects: It decreases local memory usage (less nodes in the routing tables) and it reduces bandwidth consumption due to smaller DV update packet sizes.

HSR Clustering can also be applied to proactive LSR protocols. Hierarchical State Routing (HSR) [50, 82] introduces a multilevel recursive clustering scheme: On each level, clusterheads are elected which, in turn, become members of the next higher level. The virtual connectivity[6] defines the neighborhood relation between the clusterheads on the next higher layer, which in turn dictates how higher-layer clustered are formed. The authors leave the choice of clustering algorithm open but refer to CGSR as one possible option.

In HSR, link-state information is broadcast only within a cluster. The clusterheads of each cluster accumulate the link-state information of all cluster members and distribute it only among their neighbor clusterheads. Conversely, accumulated link-state information is pushed down the logical hierarchy.

In conclusion, protocols based on clustering bypass the flooding problem by introducing a node hierarchy that *spatially* limits the flooding scope. This gain comes at the cost of *1*) increased bandwidth consumption for hierarchy maintenance (election schemes); *2*) suboptimal routes w.r.t. to the distance metric (traffic is routed over the clusterheads); and *3*) introduction of single points of failure (clusterheads) which can be a problem in

6 Virtual connectivity refers to two nodes not being within wireless transmission range but rather being indirectly connected via gateway nodes.

highly volatile or hostile environments, in case of congestion, and due to energy draining.

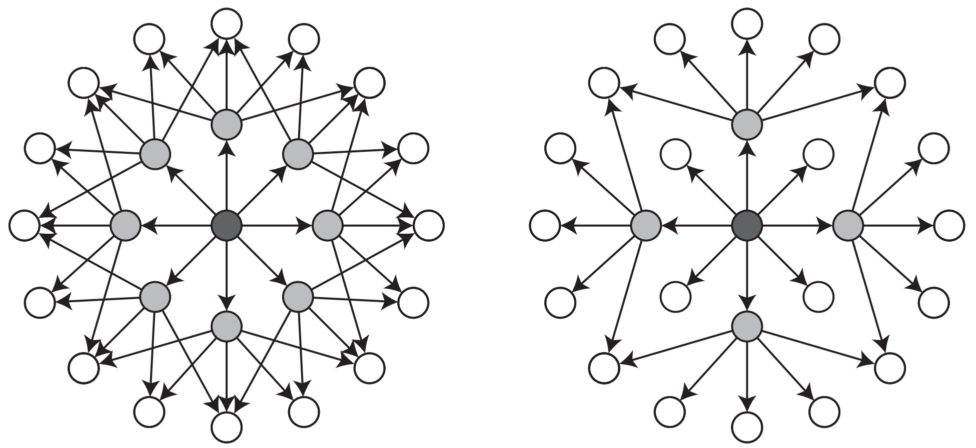
2.4.2.5 *Gossiping*

Gossiping is a probabilistic forwarding concept where each node individually tosses a coin for whether or not it should forward a message. Gossiping—an *epidemic algorithm*—was first applied to networking in 1987 for database replication [24].

Haas et al. propose gossip-based routing [40] as an extension to on-demand routing protocols such as AODV. Their idea is to probabilistically reduce the number of rebroadcasts caused by flooding. The proposed scheme is simple: A source node broadcasts a message with probability $p = 1$. Every node receiving such a message forwards it with probability $p < 1$ and discards it with $1 - p$. For the case that a source node has only few neighbors, this scheme causes the "gossip" to die early, reducing the reliability of the scheme. As a countermeasure, a forwarding probability of $p = 1$ is maintained for the first k hops. The authors show that with $p = [0.6, 0.8]$, a message is almost fully dispersed in a dense and large network (1000 nodes with 4 neighbors each), while reducing the number of rebroadcasts significantly (up to 35 %).

2.4.2.6 *Multipoint Relaying*

Multipoint Relaying [91] is a technique that minimizes the number of rebroadcasts for flooding while assuring that every node still receives the message. For multipoint relaying to work, each node selects a minimal set of its 1-hop neighbors through which it can reach all of its 2-hop neighbors. When flooding a message to the network, the message is first received by all 1-hop neighbors while only the nodes flagged as multipoint relays will rebroadcast it. The concept is visualized in Figure 2.2.



(a) Flooding without MPRs, so all 1-hop neighbors forward, yielding in a total of 8 rebroadcasts.

(b) With Multipoint Relaying, four of the 1-hop neighbors are selected as MPRs, yielding in a total of 4 rebroadcasts.

Figure 2.2: Schematic comparison of flooding without and with Multipoint Relays.

2.4.2.7 *Cognitive Routing*

Cognitive routing approaches lend themselves machine learning techniques to make routing aware of history.

SCRP An example is the Scalable Cognitive Routing Protocol (SCRP) [55]. SCRP is an on-demand protocol that aims towards limiting RREQ flooding to stable links with good radio frequencies. The resulting routes are potentially longer but comprised of better links which should lead to generally increased throughput. The protocol consists of two mechanisms: Space Flooding and Frequency Flooding. The Space Flooding protocol uses a proposed *throughput increment* metric to rate links based on their predicted capacity. The authors suggest the use of four link quality levels. When flooding, SCRP first chooses only links that are rated with a high ('4') level. When this fails, flooding starts again, but this time also allowing the flooding of links with rating '3'. As the protocol gradually includes links with a lower ranking if the destination could not be reached over higher-quality links, it eventually falls back to full flooding. This concept is similar to expanding ring search. The second protocol, Frequency Flooding, is a means to account for varying channel conditions on different frequencies in the wireless medium. Here, SCRP chooses frequencies based on packet delay, which is an "excellent metric" as the authors claim.

SCRP requires feedback from the lower layers in order to work. Unfortunately, the paper [55] does not clearly state how the throughput increment metric and packet delays are calculated.

Even though SCRP is an on-demand protocol, the basic ideas could also be applied to proactive protocols.

2.4.2.8 *Directed, GPS-aided Flooding*

GPS-aided flooding is a natural way of *spatially* limiting the search scope. The rationale is the following: When a sender already knows the geographical position of the destination, it can query a route request directed towards that destination, omitting relays that do not reside in the vicinity of the line of sight. Such protocols have the drawback of relying on the availability of geographical location information. Therefore, all protocols presented below require a mechanism to distribute location information, which in itself poses a challenge in terms of efficient dissemination. A specific distribution mechanism was described for the DREAM protocol [5] in Section 2.4.2.3. The other protocols discussed below assume the existence of an "oracle" that can be queried for the coordinates of any node.

DREAM DREAM's routing algorithm is briefly described as follows: Knowing the current trajectory[7] of the destination, the source node can compute the sector in which it expects the destination to currently reside. Packets are flooded only to nodes which are located in that sector. Only if no such neighbors exist or the routing fails, the protocol falls back to a recovery scheme, e. g., full flooding.

7 Using the periodical updates about node locations and their timestamps, the current trajectory of remote nodes can be approximated.

GPSR The Greedy Perimeter Stateless Routing (GPSR) protocol [57] choses next hops based on their geographical distance to the destination. The next hop based on *greedy forwarding* is the neighbor that is 1) the closest neighbor to the destination and 2) is closer to the destination than the forwarding node itself. If no such neighbor exists, i. e., the forwarding node is already closer than all its neighbors, the packet would be stuck at the current node. In this case, the protocol recovers with *perimeter forwarding*: The packet is routed around a face[8] using the "right-hand rule", i. e., counterclockwise along the face, until it is received by a node closer to the destination, in which case greedy forwarding is resumed.

LAR The Location-Aided Routing (LAR) [59] protocol is based on DSR. LAR introduces two schemes for selecting the geographical flooding area: The first scheme is similar to DREAM's sector calculation. It estimates the position of the destination as a circular area. The smallest rectangular area that covers both the circle and the source position is the request area which will be flooded with the RREQ. The second scheme is similar to *greedy forwarding* in GPSR. Only nodes that are closer to the destination than the previous hop will relay the RREQ, hence the request will iteratively get closer to the destination. Although LAR was presented as an extension to DSR, it is possible to deploy its concepts in other on-demand routing protocols.

2.4.2.9 *DHT-based Routing*

Distributed Hash Tables (DHTs) emerged from the area of peer-to-peer networks. A DHT can be thought of as a scalable and distributed lookup service. In a DHT, every participating node is responsible for a portion of the global ID space. The major feature of DHTs is its efficiency: querying for a certain ID has a bandwidth complexity of $O(\log n)$ (flooding would be $O(n)$), while memory consumption at a single node is low with a complexity of $O(\log n)$ (flooding would be in $O(0)$, i. e., no state needs to be stored).

AIR AND PROSE In their two 2009 papers, Garcia-Luna-Aceves and Sampath introduced two similar DHT-based routing protocols: Automatic Incremental Routing (AIR) [33] and Prefix Routing Over Set Elements (PROSE) [95]. The protocols assign *prefix labels* to each node using a distributed assignment algorithm. The prefix labels are rooted at one node in the network so that the prefix label assignment follows a tree structure of the network topology graph. Assuming that the root node has the label '0' and the protocol uses a prefix label alphabet of $\Sigma = \{0, 1, 2\}$; three child nodes receive '00', '01', and '02' as their prefix labels. In turn, node 00 assigns '000', '001', and '002' to its child nodes. This scheme is recursively applied until every node in the network is labeled.

Routing then follows the prefix labels instead of network addresses or identifiers (*NIDs*). Hashing[9] a node's NID yields a logical address in the prefix label space. The node with the longest prefix matching this hash is called the node's *anchor*. Upon joining the network, a node *publishes* its prefix label to its corresponding anchor. If a source node now wants to communicate with a destination, it needs to contact the destination's anchor node first. The source node does so by hashing the destination's NID and sending a query to that address (= the prefix label of the node's anchor). The anchor will, in

8 In graph theory, a *face* is an edge cycle surrounding a region without any edges inside that region. In this case, it refers to the area free of nodes between the node itself and the destination.
9 The authors propose a common hash function such as SHA-1.

turn, reply with the destination node's prefix label. Retrieving a node's prefix label is termed *subscribing*. In order to deal with node mobility (or topology changes in general) the protocols provide appropriate mechanisms, basically making a moving node re-join the network with a new prefix label: The incurred overhead is lower than for OLSR and AODV.

In short, the DHT-based protocols introduce a new addressing scheme, which reflects the network topology. Featuring a distributed publish-subscribe scheme, AIR and PROSE avoid flooding completely. They combine proactive and reactive schemes: *Publishing* is based on soft-state signaling (proactive) while *subscribing* is on-demand. The trade-offs are possibly suboptimal path lengths and increased load at the prefix label root.

2.4.2.10 *Network Coding*

The notion of *Network Coding* was first introduced by Ahlswede et al. [2] for point-to-point multicast packet networks. The basic idea is to maximize the information flow between a source and multiple receivers. This is achieved by coding packets together at intermediate nodes before forwarding their information to the next hop.

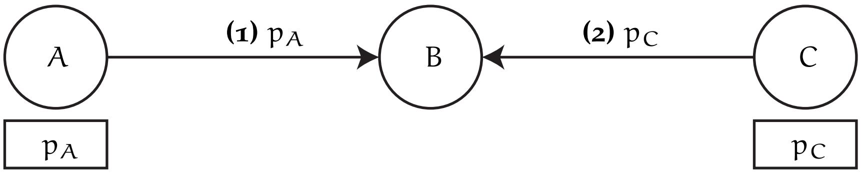
COPE In 2006, the first practical network coding-based forwarding scheme, COPE, was proposed by Katti et al. [58]. COPE uses network coding to reduce the number of transmissions by exploiting the broadcast nature of the wireless medium. In contrast to the work of Ahlswede et al., COPE is designed to improve *unicast* rather than multicast routing. The basic concept is illustrated in an example (Figure 2.3): Assume a simple network consisting of nodes A, B and C. Assume further that A wants to communicate with C and at the same time C wants to transmit a packet to A, while B serves as a relay. In traditional forwarding, four transmissions are required to deliver a packet from A to C and vice versa (A to B, C to B, B to C, and B to A). In COPE, only three transmissions are necessary: After A and C have sent their packets, p_A and p_C, to relay B, B computes a combined packet $p_{AC} = p_A \oplus p_C$ of equal length[10]. p_{AC} is then transmitted (broadcast) to both nodes, A and C. Using its information about p_A, A is able to decode C's packet as $p_C = p_{AC} \oplus p_A$. Decoding of p_A at C is performed analogously.

COPE operates in an opportunistic manner. It codes only packets together that are in a node's local queue. It does not wait for packets to arrive, and thus does not introduce a significant forwarding delay.

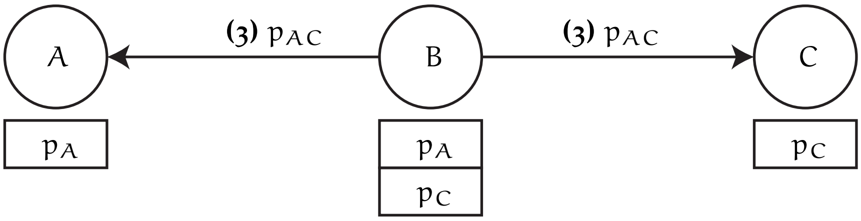
Assume that a node has received packets $p_1, p_2, \ldots, p_k$ that it needs to forward to nodes $n_1, n_2, \ldots, n_k$, respectively. If the node can be sure that every node n_l has already "seen" all packets $p_m, m \neq l$, then it can serve all recipients with a single packet $p = p_1 \oplus p_2 \oplus \cdots \oplus p_k$. Hence, the throughput gain for a single coding opportunity is—in theory—unlimited. In order to determine which neighbors have already seen which packets, COPE relies on three mechanisms: *1*) Periodic broadcasts, where each node lists hashes of seen packets; *2*) if node n_i sent a packet p to n_j, then n_j knows that n_i has seen p; *3*) if n_j cannot be confident using the previous two mechanisms, it computes a probability that n_i has also received p from another neighbor.

COPE does not rely on a specific routing protocol, i. e., *how* next hop decisions are made. As such, COPE can be seen as an orthogonal scalability-improving approach to

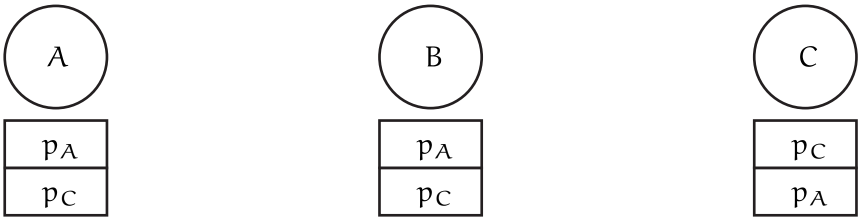
10 The $\oplus$ operator stands for a bitwise XOR operation.



(a) Nodes A and C send their packets to the relay B.



(b) Relay B broadcasts the coded packet p_{AC}.



(c) A and C have both received p_{AC} and decoded p_C and p_A, respectively.

Figure 2.3: Demonstrating the core concept of COPE with a simple topology and a total of three transmissions (compared to four with traditional forwarding).

routing and could consequently be applied to any routing protocol presented in this chapter.

Sengupta et al. term a routing protocol that uses coding as a forwarding mechanism but otherwise does not make choices based on coding opportunities as *coding-oblivious routing*. In their paper [99], they promote *coding-aware* routing protocols because throughput gains can be substantial: up to 40 % over coding-oblivious routing depending on the network topology and traffic patterns. Such improvement is achieved by maximizing coding opportunities while minimizing interference. A practical evaluation is missing since the paper is of a theoretical nature.

2.4.2.11 *Alternative Routing Metrics*

The goal of a routing protocol is not only finding any route but also selecting the "best" one when there are multiple options available. Most popular MANET routing protocols rely on an Internet-like [71] routing metric, i. e., shortest-path, as a selection criterion. *Shortest path* can be defined as fewest number of hops, most available bandwidth or lowest delay. These performance-related metrics can fluctuate dramatically in a mobile environment causing frequent route re-discoveries.

Continuously Adapting Secure Topology-Oblivious Routing (Castor) [31, 30], for example, uses reliability estimators as a primary metric for next-hop selection. The rationale is simple: reliable routes are stable, and there is less need for route repairs[11].

11 The reliability estimators are also key ingredients for Castor's excellent performance as a *secure* routing protocol, which will be discussed in Section 3.

In DTNs, the notion of shortest paths does not even exist due to the potential lack of available paths altogether. As a result, the DTN research community is forced to investigate alternative metrics for selecting next hops [100]. Most DTN metrics are *destination-dependent*, i. e., are based on the likelihood of a node meeting a specific destination (history of last encounters, social networks, etc.). The one-to-one applicability of these metrics to MANETs is limited, but the ideas can be transferred. Instead of using the history of last encounters (which is correlated to the probability of that node meeting the destination again), we can use delivery probabilities of a node to a destination based on past behavior as in Castor. Other metrics are *destination-independent*: Movement patterns could be used to determine the likelihood of a node remaining within transmission range and hence being able to continue to act as an next hop. This way, a node can proactively switch to another route before packets are dropped and would have to be retransmitted. Similarly, node resources can affect the suitability to act as a relay: If a node's battery is to drain out soon, it should be avoided as a next hop.

In conclusion, alternative routing metrics can improve scalability by avoiding frequent route repairs and thus flooding, especially in highly dynamic networks. Destination-independent metrics can be used as a secondary metric if multiple paths to a destination are known.

2.4.2.12 *Infrastructure Support*

MANETs do not rely on a predefined communication infrastructure. However, they could make opportunistic use of such infrastructure when available.

A practical example for infrastructure-supported mobile networks are Vehicular Ad-Hoc Networks (VANETs). Such networks consist of mobile nodes (the vehicles) and also fixed infrastructure nodes (roadside equipment). One scenario is that of vehicles communicating with each other to avoid large rear end collision by transmitting breaking signals to other proximate cars whose drivers cannot see the breaking lights ahead yet. Roadside equipment can provide the mobile nodes with additional information, such as upcoming roadwork or traffic jams, or they could even provide uplinks to the Internet.

In summary, infrastructure nodes can provide information that otherwise would have to be retrieved using flooding. However, their applicability is limited to special scenarios. Thus, in this thesis, we will not consider infrastructure support as a means for improving the scalability of a MANET routing protocol.

2.4.2.13 *Multipath Support*

Most protocols such as for example AODV and DSR are designed to find (at most) *one* route to a destination. However, due to the unreliable nature of MANETs, knowing (and concurrently using) multiple routes to a destination can result in better scalability. Multiple paths can provide load balancing, fault-tolerance (redundancy, thus less retransmissions) and a higher aggregated bandwidth (helps to avoid congestion) [72]. Depending on the expected reasons of packet loss, paths should be either *link-disjoint* or *node-disjoint*, which provide resilience against link failure or node (and link) failure, respectively.

SMR Split Multipath Routing (SMR) [64] is based on DSR. In contrast to DSR, SMR does not discard duplicate RREQs at intermediate nodes if they are received over different links. The destination will immediately answer the first RREQ it receives (a RREP is sent back to the source) to minimize route discovery delay. It then waits for other incoming RREQs for a certain period of time. From all further RREQs, it selects the $k-1$ (k being the desired number of paths) routes that are *maximally link-disjoint*[12] to the route with the shortest delay. The authors set $k = 2$ in the paper, i. e., only two routes are discovered.

AOMDV Ad hoc On-demand Multipath Distance Vector (AOMDV) routing [69] extends AODV with multipath routing. It aims to discover node-disjoint paths, but also accepts (the weaker) link-disjointness if not enough node-disjoint paths exist. AOMDV includes the first hop (a neighbor of the source) in each RREQ. Forwarding nodes record neighbors that sent a RREQ with an unseen "first hop" field. However, only the first received RREQ is forwarded. This way, each node maintains a set of node-disjoint paths to the source. A destination replies to a RREQ with up to k RREPs (k indicates the desired number of paths, but is bound by the number of neighbors from which it received a RREQ). When an intermediate node receives more than one RREP, it will forward it over node-disjoint paths as recorded from the received RREQs.

2.4.3 *Discussion*

In the following, we discuss applicability and compatibility of the scalability improving mechanisms. The questions we try to answer are: Which mechanisms can be applied to which type of general routing protocol? Which mechanisms can be combined in order to achieve even better scalability? Which mechanisms are universally applicable? Which combinations might even be harmful to routing performance?

2.4.3.1 *Applicability*

Table 2.1 summarizes the applicability of the various mechanisms to the different types of routing principles introduced in Section 2.4.1, i. e., proactive vs. reactive routing, and the Distance Vector Routing (DVR), Link State Routing (LSR) or Source Routing (SR) protocols. We denote the applicability as either yes (■) or no (□). We note that clustering-based protocols follow a proactive strategy as they need to exchange information to maintain the hierarchy. We further note that DHT-based routing is deemed not applicable to any of the standard routing protocols as DHTs fundamentally change the routing process (new identifiers, distributed lookups, implicit routing based on the identifiers).

2.4.3.2 *Compatibility*

Below, we discuss the compatibility of the various mechanisms from the previous section. Table 2.2 is meant as a visual guidance to the reader.

As a first remark, some are mechanisms are incompatible (■) with each other, e. g., the ones for reactive and proactive routing. Others, such as route caching and expanding ring search can be a powerful combination: If appropriately distributed through the network, route caches increase the probability that expanding ring search experiences a

12 The authors do not clearly state whether they attempt to create link-disjoint or node-disjoint paths. However, since RREQs are forwarded multiple times, the forwarder is contained in different reverse paths and so the returned paths will most likely not be node-disjoint.

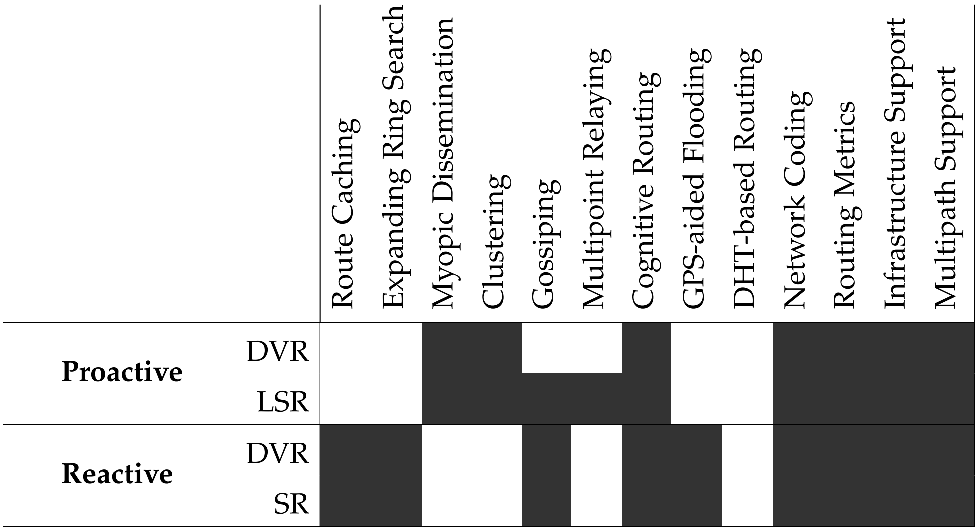


Table 2.1: Applicability of unicast routing schemes.

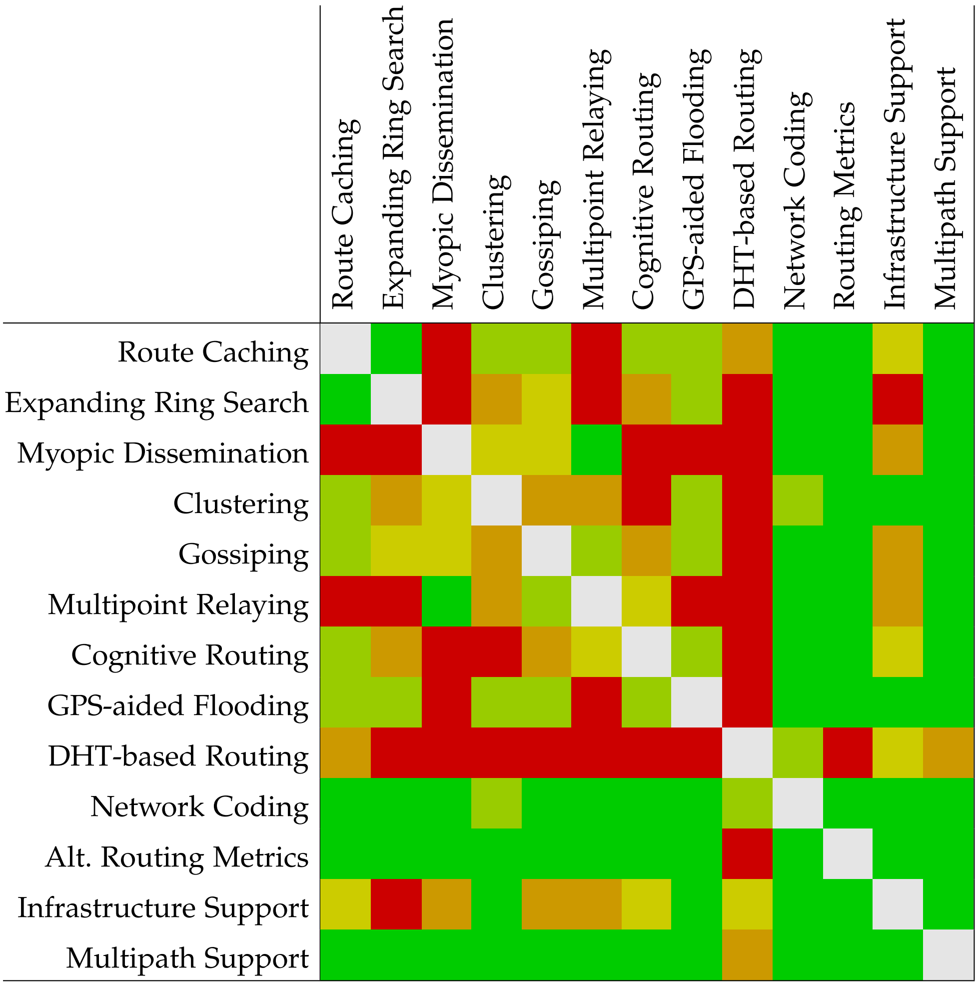


Table 2.2: Compatibility of unicast routing schemes.

cache hit in an early iteration (■). Most of the combinations are neither extreme, i. e., mechanisms that can be deployed in combination but might yield in different levels of scalability gain (■, ■, ■, from *high* to *low*).

Myopic dissemination, gossiping and multipoint relaying are all mechanisms that aim for scoped flooding. Even though gossiping was introduced as a means to improve on-demand routing, the concept can easily be applied to proactive dissemination of link-state information. As a result, they are compatible with each other. However, in combination, they might limit flooding too much so that network-wide packet dispersion is hardly possible. We do not see how these three approaches can be improved by infrastructure support (■).

Cognitive routing can be combined with other mechanisms but the improvement is questionable. Expanding ring search, for example, limits the search space to a circular area while the cognitive protocols restrict the search space to reliable links. In combination, the cognitive scheme might downgrade its link quality requirements because RREQs might not reach the destination due to the hop count limit. The same reasoning applies to gossiping (■).

GPS support can basically enhance any on-demand protocol with directional search. It cannot improve (proactive) LSR since LSR requires network-wide flooding. Unfortunately, the usage of GPS requires a means of distributing initial location information among network nodes which leads to similar problems already encountered in routing, i. e., how to distribute information *efficiently*.

Network Coding is shown to be universally applicable. This is due to the fact that it does not interfere with routing decisions but only affects packet forwarding (■). Combining DHTs and Network Coding might yield in limited gain compared to other schemes with Network Coding: with DHTs, there will be no flooding, thus less coding opportunities might arise (■).

DHT-based protocols are, in contrast, basically incompatible with most other mechanisms (■). This arises from the fact that DHTs abandon flooding altogether, which is what other mechanisms try to improve. Route caching seems to be a compatibility candidate. However, since DHT lookups are already very cheap in terms of communication cost, caches can only marginally improve performance. On the contrary, caches might actually degrade performance due to outdated information (■). DHTs cannot be used in conjunction with alternative routing metrics either since next hop selection is dictated by the prefix labels.

We realize that least-hop-count routing metrics are not necessarily best-suited for application in MANETs. Alternative metrics can actually improve the throughput of the discovered paths [19]. With the exception of DHTs, alternative routing metrics can be applied to virtually any reviewed mechanism. For example, cognitive and GPS-aided routing protocols already apply alternative routing metrics: reliable links and geographical distance to a location, respectively. Schemes with infrastructure support can benefit from alternative routing metrics, e. g., if infrastructure nodes are more powerful and reliable than other nodes in the network, a routing metric could give priority to routing over these nodes (■).

Infrastructure support has a potentially positive effect in schemes where certain nodes take special roles, e. g., in clustering. Infrastructure nodes could improve the performance of cognitive protocols because they could increase the reliability of chosen paths (assuming that infrastructure nodes are more reliable than regular nodes). In GPS-supported schemes, infrastructure nodes could act as location databases which can be queried for coordinates of remote nodes. This could solve the problem of disseminating the location information as mentioned earlier.

Multipath support is another approach that can be applied to a variety of routing protocols. The only requirement for the routing protocol is to return more than one route to a destination. Enabling DHT-based protocols PROSE and AIR with multipath support is not trivial since the routing approach is based upon the prefix labeling scheme (■).

2.5 scalability of manet multicast routing

In this section, we discuss the fundamentals of *multicast*, i. e., one-to-many communication, and present techniques for improving the scalability of multicast routing protocols in MANET environments. Similarly to Section 2.4, we conclude with a discussion of the proposed solutions.

2.5.1 *Fundamentals*

Multicast is used to address several recipients with a single message. In a simple solution, an existing unicast protocol is used to transmit the same message multiple times, one for each intended receiver. This approach is highly inefficient, especially for applications such as live-TV streaming, because the same data would be transmitted multiple times over the same link(s). Dedicated multicast protocols remove this redundancy, thus improving the scalability of such applications. In order to efficiently address multicast *groups*, i. e., the group of intended receivers, various mechanisms have been proposed.

2.5.1.1 *Stateful vs. Stateless Multicast*

We differentiate two general types of multicast protocols: *stateful* and *stateless* [23].

Stateful Multicast. Several MANET protocols adopt the concept of Internet multicast protocols. The basic idea is to construct an optimal distribution graph, which covers all group members and can then be used for communication. This can be seen as the equivalent of route discovery for unicast routing protocols. Typically, these distribution networks have a tree or mesh structure. However, hybrid approaches have been proposed as well.

Tree-based. Tree structures are—from a graph theory point of view—the most efficient structure for message delivery. For example, if we create a minimum spanning tree covering all group members, the number of links used for dispersion of a multicast message should be minimized. However, in a volatile environment with high mobility, having only single paths to each destination is insufficient for robust operation [65].

Trees can either be source-initiated, i. e., the source acts as the root of the delivery tree, or shared (core-based [4]), i. e., some central node at a rendezvous point acts as the root.

The Multicast Ad Hoc On-Demand Distance Vector (MAODV) protocol [94] is an example for a tree-based protocol. It creates multicast trees through RREQ flooding, directly following the concept of AODV. Each receiver replies with a unicast RREP to indicate its desire to join the multicast group. MAODV follows a hard-state approach, meaning that topology and membership changes have to be actively detected and mitigated.

Mesh-based. In the MANET domain, exploiting the mesh structure of the network can improve the robustness of multicast. Typically, several routes to any group member are available. By maintaining alternative paths, mesh-based schemes are able to deliver packets even if individual links fail. These approaches can be seen as the multipath supporting equivalent of unicast routing.

Unlike MAODV, On-Demand Multicast Routing Protocol (ODMRP) [66] uses a soft-state approach for group membership. As long as a source wants to transmit data, it floods the network with Join Queries indicating its presence. If the query is received by a group member, it broadcasts a Join Reply message. This way, multiple paths from source to receivers are set up (in contrast to MAODV where single routes are set up). Periodic flooding assures that topology changes are implicitly mitigated.

The major drawback of ODMRP is that it requires periodic flooding[13] of Join Queries, leading to scalability problems.

Stateless Multicast. Apart from the traditional, stateful approaches, some research has gone into investigating *stateless* multicast. The rationale behind this is the avoidance of expensive group management (tree or mesh) which can cause excessive overhead [37]. Instead, those protocols rely on unicast routing.

A specific class of stateless multicast schemes is *Explicit Multicast (Xcast)*. In Xcast, sources list all destinations explicitly in the header of every packet to the group. This implicates increased bandwidth usage because the header size grows linearly with the group size. This is why Xcast is intended for small groups, and hence sometimes referred to as *Small Group Multicast (SGM)* in the literature.

We have to refine our definition of *scalability* for multicast: Stateless multicast protocols do not scale well with large group sizes; however, a large number of frequently-changing groups are better supported. On the other hand, stateful approaches are designed to support larger groups but maintenance overhead increases with the number of groups in the network. Frequent membership changes incur additional overhead. We summarize: Multicast protocols can *1*) *scale with the group size* or *2*) *scale with the number of multicast groups in the network*.

2.5.1.2 *Overlay Multicast*

Overlay multicast builds a virtual topology layered over the physical MANET topology, e. g., AMRoute [105]. The edges of an overlay multicast tree are tunneled unicast links: Multicast traffic is sent from group member to group member using some underlying unicast routing protocol. This provides the following advantages: *1*) The virtual topology does not need to change, i. e., it can remain static because physical topology changes are handled by the underlying unicast protocol. *2*) Nodes not interested in multicast

13 while sending

communication do not need to support the multicast protocol since it is encapsulated in the unicast traffic.

However, this comes at cost of inefficiency and increased delays: Since the tree construction itself is oblivious to the physical network topology, neighbors in the virtual topology might in fact reside in distant parts of the physical network. This leads to packets being routed across the network. Furthermore, different tunnels might share the same physical links which leads to redundant transmission of the multicast packets—diminishing what multicast set out to improve.

2.5.2 *Improving Scalability*

A plethora of work has been carried out to improve the performance of MANET multicast routing [23, 56]. Here, we extract some of the core concepts of these protocols.

2.5.2.1 *Multiple Core Based Tree*

The notion of Core Based Trees (CBTs) was introduced for multicast tree creation in wired networks [4]. The idea is to have *one* shared tree created for the entire multicast group, i. e., not every sender is required to set up its own tree. This is done by selecting a single router as the "core" of the tree. The core presents itself as a single point of failure for the multicast group. This makes traditional CBTs inappropriate for the MANET domain where the likelihood of node failure is high.

CAMP Core-Assisted Mesh Protocol (CAMP) [32] tackles this problem by using multiple cores. The cores are used as landmarks for joining nodes and thus avoid flooding of the network with control messages (such as in ODMRP). Since CAMP uses a mesh for transmitting multicast packets, it is able to cope with node mobility or link breakage.

CAMP assumes that cores are statically pre-configured and does not provide a dynamic core selection algorithm.

2.5.2.2 *Soft-State Forwarding Group*

The Forwarding Group Multicast Protocol (FGMP) [16] introduces the concept of Forwarding Groups (FGs). Multicast packets are solely forwarded by members of the FG. This makes FGs implement a sort of scoped flooding. Selection of FG nodes is based on either receiver or sender advertisement. In receiver advertisement, all receivers periodically advertise their membership information in the network. The sender receives all advertisements, computes a forwarding table from it and sends this information to all its neighbors. The neighbors, in turn, determine for which nodes they act as forwarders. Sender advertisement works the other way round, letting senders periodically indicate their presence.

FG nodes have an expiration timer for every member they forward packets to. If this timer expires (and no update was received in the meantime), the member is removed from the table. This soft state approach makes FG suitable for MANET environments.

ODMRP uses the Forwarding Group concept to set up the meshes for every group.

2.5.2.3 *Directed, GPS-aided Multicast*

GPS can be used to improve the creation of multicast delivery trees. The same general drawbacks as for GPS-aided unicast routing apply here, namely the problem of location information distribution in the network.

GEOCAST Geocast [48] is a well-known example for GPS-based multicast. It differs, however, from the other multicast mechanisms discussed in this section. Geocast is used to address a geographical *area*, or more precisely, all nodes that reside within the addressed area. While useful for some applications, it is not possible to address an arbitrary group of nodes that is scattered around the network with Geocast. Due to this limitation, it is not further discussed, but mentioned here for the sake of completeness.

LGK AND LGS Chen and Nahrstedt propose two location-guided overlay multicast tree construction algorithms: location-guided k-ary (LGK) tree and a location-guided Steiner (LGS) tree [14]. Both algorithms attempt to create least-cost delivery trees based on geographic distances and a greedy heuristic. The proposed schemes are stateless in the sense that the source lists the receivers in each data packet header. Upon reception, each node locally decides where to forward the packet to based on geographical closeness. In LGK, each node selects the k closest next hops. LGS, in contrast, creates delivery trees with variable fan-out. According to the authors' simulation results, LGS incurs lower bandwidth cost than LGK.

Overlay trees usually have the problem of creating topology-oblivious trees, which makes multicast routing inefficient. However, by using location information, LGK/LGS become somewhat aware of the physical topology, thus mitigating the problem.

2.5.2.4 *Hierarchy*

HIERARCHICAL SGM Gui and Mohapatra propose a hierarchical multicast scheme that allows for applying SGM to larger groups [37]. The basic idea is to partition the multicast group into smaller and better manageable subgroups. Each subgroup selects a head node that will be part of a higher level group. The approach is similar to the concept of HSR for unicast routing (Section 2.4.2.4).

E2M Extended Explicit Multicast (E2M) [35] introduces the notion of Xcast Forwarders (XFs). The idea is based on the observation that all group members can be reached over just a few outgoing links. This means that a single neighbor might be responsible for forwarding messages to a larger portion of the group. XFs announce themselves as proxies to a subgroup (or subtree) of the forwarding tree. This way, the source only needs to include a few XFs in the header in each message. When an XF receives a multicast message, it will expand the destination list in the header by the downstream group members.

Each node can autonomously decide whether to become an XF. The authors propose a selection strategy that is based on *1*) the number of downstream group members, and *2*) whether the node itself is a branch in the tree.

2.5.2.5 *Membership Caching*

The main drawback of Xcast is the inherently large header size (all members are listed in every packet header). Different approaches for reducing the header size have been proposed to improve the performance of Xcast in larger groups. One solution lets intermediate nodes cache group memberships.

DDM The Differential Destination Multicast (DDM) [52] protocol is, to our knowledge, the first concrete suggestion of stateless multicast for MANETs. DDM attempts to circumvent the drawback of increased header sizes by introducing a "soft-state" operation mode. In this mode, nodes remember where multicast packets for a particular session were routed to the last time. Thus, the complete destination list does not need to be included in every packet: After a full list has been sent, only changes in the destination list or routing tables are encoded in the header. Since nodes might miss some updates, the full list should occasionally be included in the headers.

The authors propose the soft-state mode in networks with relatively small number of multicast groups such that state information stored at every node is kept low.

2.5.2.6 *Gossiping*

RDG Route Driven Gossip (RDG) [68] is a stateless multicast protocol. In RDG, every group member has a (partial) view on the multicast group membership, i. e., every node knows at least some other nodes that are in the same group. RDG is oblivious to the network topology (overlay routing) and implements a random *infection* of nodes. The basic idea of RDG's packet distribution is as follows: *1*) The source chooses a random subset of the group members from its view; *2*) it transmit the message to every chosen group member using an underlying unicast routing protocol; *3*) the receivers are now *infected* and in turn select subsets from their views; *4*) the procedure continues until the message has been dispersed in the multicast group.

Since RDG is oblivious to the network topology, the random transmission of messages across the network is inefficient. An example protocol run in the authors' paper[14] reveals that it can degenerate to full flooding, meaning that almost all nodes in the network eventually participate in packet forwarding in at least one of the iterations. Therefore, the authors propose an optimization for RDG such that path lengths are considered when choosing the group member subset for forwarding. This way, nodes that are closer to the transmitter will be chosen with a higher probability.

According to the authors' simulation results, RDG is able to operate sufficiently reliable even under high node mobility.

2.5.3 *Discussion*

We discuss applicability and compatibility of the scalability improvement mechanisms for multicast, similarly to Section 2.4.3. Analogously, we try to answer the following questions: Which mechanisms can be applied to which type of general routing protocol? Which mechanisms can be combined in order to achieve even better scalability? Which mechanisms are universally applicable? Which combinations might even be harmful to the routing performance?

14 Figure 5 of the RDG paper [68]

2.5.3.1 *Applicability*

Table 2.3 summarizes the applicability of the mechanisms presented in the previous subsection to the basic multicast schemes. We denote the applicability as yes (■) and no (□). Multiple Core Based Tree and Soft-State Forwarding Group are concepts to improve tree-based and mesh-based protocols, respectively. GPS support can be added to any multicast type, while hierarchies, caching and gossiping are all stateless approaches.

2.5.3.2 *Compatibility*

In the following, we discuss the compatibility of scalability improvement mechanisms for multicast. Table 2.4 acts as a visual guidance for the reader. We look at stateless multicast approaches only, since we identified only a single mechanism for each of the two stateful approaches, i. e., tree-based and mesh-based protocols.

Hierarchies and membership caching both attempt to reduce the header size of Xcast messages for improving the scalability within larger groups. Caches help to further reduce the header size of Xcast packets. However, if the network exhibits high mobility, caches will often contain outdated information, leading to packet loss. Thus, deciding for the use of caches should be carefully weighed up (■). Hierarchies are based on the idea of header expansion at intermediate nodes. Xcast Forwarders appear to be better suited for highly dynamic networks since intermediate nodes can autonomously decide whether to be come a proxy or not, depending on the stability of downstream members. A multi-level hierarchy requires more coordination among nodes so maintenance may become an overhead problem if the topology changes often; however, they are able to support even larger groups than XFs.

Gossiping also achieves a reduction in header size. It is based on the assumption that membership information is not completely available at the source but instead probabilistically shared among all group members. Thus, the delivery scheme is different from the other two stateless approaches and compatibility low (■). GPS information could be used to choose next hops in the topology-aware variant of RDG.

Geographical information could further facilitate the creation of hierarchies (■). In general, location information would be helpful to support directed flooding, especially for the stateless approaches (■).

		Multiple Core Based Tree	Soft-State Forwarding Group	GPS-aided Multicast	Hierarchy	Membership Caching	Gossiping
Stateful	Tree-based						
	Mesh-based						
Stateless							

Table 2.3: Applicability of multicast routing schemes.

	Multiple Core Based Tree	Soft-State Forwarding Group	GPS-aided Multicast	Hierarchy	Membership Caching	Gossiping
Multiple Core Based Tree						
Soft-State Forwarding Group						
GPS-aided Multicast						
Hierarchy						
Membership Caching						
Gossiping						

Table 2.4: Compatibility of multicast routing schemes.

3 SECURITY OF MANET ROUTING

In this chapter, we summarize and discuss literature work towards securing MANET routing.

3.1 DEFINITION

Buttyan and Hubaux [11] define three fundamental security operations for Wireless Mesh Networks: *1*) securing routing, *2*) enforcing fairness, and *3*) detecting corrupted nodes.

In this thesis, we will refer to *secure* Mobile Ad-Hoc Networks as systems implementing the first operation, securing routing. From the six security goals authentication, access control, confidentiality, integrity, non-repudiation [49], and availability [102], we consider only *authentication*, *availability*, and *integrity*[1]. Fairness/quality of service (QoS) and detection of corrupted nodes is out of scope of this thesis.

We can break down the task of securing MANET routing into securing the three core components: *neighbor discovery*, *route discovery* [11] and *data transmission* [80]. Neighbor discovery is used to find other nodes within direct transmission range. Route discovery is the process of finding routes, i. e., the mechanisms discussed in Chapter 2. When route discovery succeeds, nodes can send the actual user data (data transmission). We consider a routing protocol secure if it remains operational even when attacked on any or all components. For example, if neighbor discovery fails, then route discovery might fail as well if it relies on correct information about the neighborhood. A protocol that is able to securely discover routes, i. e., implements secure neighbor and route discovery is of no use if the actual data transmission remains insecure: a malicious node might play along in the route discovery phase but start dropping data packets later.

In the following, we refer to an *adversary* as an entity that wishes to impair MANET routing (on any component). A *malicious node* is a node controlled by an adversary; an adversary can control multiple nodes. An *attack* is a method to achieve an adversary's goal, e. g., Denial of Service (DoS).

3.2 ASSUMPTIONS

Security, especially in MANETs, is a complex issue. We state assumptions on the system and define adversary capabilities in order to define the scope of this chapter.

3.2.1 *Secure Neighbor Discovery*

Neighbor discovery is used to find other nodes within direct transmission range. Some protocols or applications require the existence of such a component in order to be secure.

1 Access control and non-repudiation are only meaningful on the application layer. Confidentiality could be achieved on the network layer to secure higher-layer protocol traffic which is not considered a necessity here.

Secure neighbor discovery is usually implemented by a *distance bounding* protocol such as [10]: Nodes try to estimate the physical distance to some other node using the signal propagation delay between them. Only if the estimated distance is below a certain threshold, a node is deemed a neighbor. Poturalski et al. propose a framework towards provably secure neighbor discovery protocols [89].

In this thesis, we only consider secure route discovery and data transmission: In-depth discussion of secure neighbor discovery mechanisms is not considered since a secure routing protocol can be built without the assumption of a secure neighbor discovery protocol, as we will show in Chapter 4.

3.2.2 *Key Distribution and Management*

All secure routing protocols presented here rely on Security Associations (SAs) between certain nodes, e. g., end-to-end. SAs can be based on two types of keys:

Public keys. Every node in the network has a public key which has to be known by the communicating party and vice-versa.

Symmetric keys. Every communicating node pair has a shared secret that is exclusively known to them.

Computational overhead for symmetric cryptography is lower than for asymmetric cryptography. However, SAs based on public keys are cheaper to set up in terms of bandwidth consumption. A key distribution and management facility is required in either case. Both enumerated approaches (pre-distribution and on-demand distribution) exhibit their own scalability trade-offs.

Pre-distribution. Before joining the network, nodes are supplied with the required keys, e. g., public keys of all other participating nodes. This is a practicable approach if all nodes are known beforehand. In self-organizing networks, however, it lacks the flexibility to accommodate new nodes and to safely remove old nodes (*key revocation*).

On-demand distribution. If keys are not pre-distributed, we can either

- query a (designated) network entity which is responsible for key management, e. g., a Certificate Authority (CA) (single point of failure), or
- send signed keys opportunistically in the routing packets (overhead increases linearly with route length).

Designing secure and reliable on-demand key distribution and management for MANET environments is a challenging topic by itself and has acquired substantial research interest (e. g., [104]). The issue of key revocation persists here as well.

In this thesis, we leave aside the problem of key distribution. We assume that all required keys are pre-distributed and SAs are readily set up.

3.2.3 *Adversary Model*

We now describe our adversary model, which is based on Buttyan and Hubaux [11] and Galuba et al. [31], and which will be used in the following discussions. We basically assume a weaker variant of the Dolev-Yao "man in the middle" adversary [26]. Specifically, the adversary

- can be a valid member of the network taking part in the protocol execution (*internal*) or not (*external*);
- can interfere with the protocol operation by message manipulation or forgery (*active* or *Byzantine*) or just eavesdrop on the communication (*passive*);
- can control only a portion of the communication links, e. g., by controlling a portion of the participating nodes. Control of *all* links as in the Dolev-Yao model is too strong since we are concerned with availability. If an attacker had access to *all* links then there would be no reasonable way to thwart Denial of Service attacks;
- cannot break cryptographic primitives.

The strongest model is the *internal active* adversary, which is able to conduct all attacks presented below. The weakest model is *external passive*. Note that we do not focus on a fully *passive* adversary who could perform traffic analysis. Protection mechanisms against this kind of attack are beyond the scope of this thesis.

3.3 SECURITY OF MANET UNICAST ROUTING

In this section, we summarize popular attacks on the network layer, particularly on route discovery and data transmission. Recall that we are not concerned with attacks on neighbor discovery. We focus on attacks that tamper with routing itself, e. g., attempt to produce fake routes. Other attacks such as RREQ flooding achieve DoS by brute-force resource consumption, do not target the correctness of the routing protocol. Rate limiting schemes can be applied to mitigate such attacks. An extensive survey of attacks on MANETs covering all layers was conducted by Sen [98].

Based on the attacks described, we then review selected MANET routing protocols that have been designed with security in mind. We briefly describe a number of protocols that we consider exemplary for the academic research that has been carried out on secure route discovery and data transmission, respectively.

3.3.1 *Attacks*

Attacks on route discovery aim towards controlling discovered routes, i. e., placing malicious nodes on routes that the adversary wishes to control. Such an adversary could then, for example, start a passive traffic analysis attack or stop forwarding data packets on that route (DoS).

There may be several other subtle attacks based on the interaction of the routing protocol with a higher-layer transport protocol, e. g., the Jellyfish attack [1] on the congestion control mechanism of the Transport Control Protocol (TCP) [88]. Since we do not consider a transport protocol in our model, such attacks shall not be a focus in this thesis.

3.3.1.1 *Spoofing Attack*

The spoofing attack is based on fabricating routing information, i. e., RREPs or LSUs. Sources in on-demand protocols usually use the first RREP received for route selection. So, if a malicious node directly replies to a RREQ and no benign node has a fresh route to the requested destination, chances are high that the adversary's RREP is received before any other (legit) reply. This attack basically "attracts" traffic towards the malicious node.

3.3.1.2 *Sybil Attack*

The Sybil attack [27] is not specific to MANETs but to distributed systems in general. The basic idea is that a single node has multiple virtual identities so that the relationship between entity and identity is one-to-many instead of one-to-one which would be the usual case. The credentials (key material) for the identities could be taken from compromised nodes.

The consequences of a successful Sybil attack depend on the system being attacked. For example, if a system deploys a reputation scheme where each node is rated based on its forwarding reliability, a Sybil adversary can switch between its multiple identities to elude bad ratings. Another example is multipath routing: the adversary masquerades as different nodes on each path during route discovery. This gives it full control over the transmission, thus, jeopardizing the sought-for robustness of node-disjoint paths.

3.3.1.3 *Rushing Attack*

Rushing attacks [44] target on-demand protocols. During route discovery, intermediate nodes only relay the *first* RREQ they receive in order to minimize the flooding impact. This means that if a node is *fast* in relaying the RREQ, it will be likely included in the discovered route. To be faster than non-attacking nodes, the adversary ignores delays that are typically used at the link and network layers for collision avoidance.

3.3.1.4 *Wormhole Attack*

A wormhole attack (Figure 3.1) is more sophisticated than a rushing attack and requires two or more colluding malicious nodes. A *wormhole* is set up using high speed (possibly out-of-band, wired or directed RF) links between the colluding nodes. RREQs from benign nodes are forwarded along those high speed links. Thus, route discovery will most likely return a route containing the wormhole link if a shortest path routing metric (Section 2.4.2.11) is used for route selection. This way, an adversary can control large parts of the global network traffic using few, properly placed nodes.

Wormholes are hard to detect because they do not forge RREQs or RREPs which could be thwarted with authenticity checks. Wormholes are hardly distinguishable from very fast but valid links. Options to counter wormhole attacks include the use of geographical or temporal packet leashes [43].

3.3.1.5 *Tunneling Attack*

A tunneling attack is conceptually similar to a wormhole attack. However, in this case, the colluding nodes do not use out-of-band links to communicate. Instead, RREQs and RREPs are encapsulated in new data packets and transmitted between the colluding

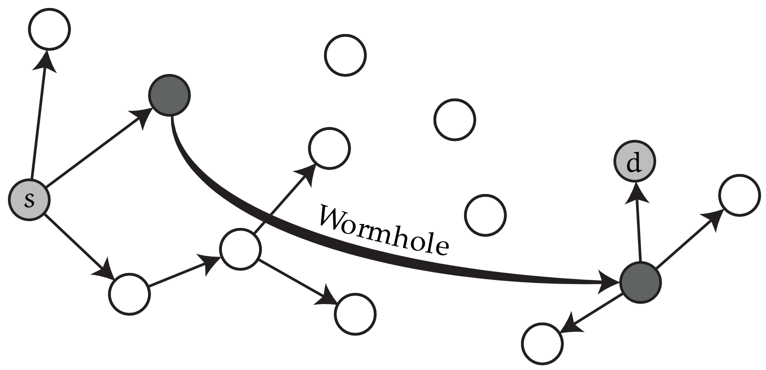


Figure 3.1: Example of a wormhole attack: Source s reaches destination d with only 3 hops.

nodes using the existing network. This makes the attack weaker than the wormhole attack in the sense that transmission of (encapsulated) packets cannot be faster than the actual network permits. However, the target is different to wormhole attacks: *Tunnels* attack the topology metric (such as hop counter) of RREQ or LSU packets. Since packets are packed on one side of the tunnel and unpacked at the other side, the hop counter of the original packet is not changed during transit. This deceives a receiver into believing that the source is much closer (w. r. t. the topology metric) than it really is. Consequently, the attack only affects protocols based on such a topology metric (packet delay cannot be attacked by tunneling).

3.3.1.6 *Blackhole Attack*

If a malicious node has placed itself on some forwarding path (either because it has compromised route discovery or was legitimately selected as part of the route), then it may conduct a blackhole attack. A blackhole (or sinkhole) adversary simply drops all packets that it is supposed to forward, resulting in DoS.

3.3.1.7 *Grayhole Attack*

A grayhole attack is a variation of a blackhole attack in which the adversary drops packets selectively, either *1*) for certain nodes only, *2*) for a limited amount of time only, or *3*) a combination of both.

This adversarial behavior is hardly predictable, making grayhole attacks difficult to detect since selective drops could also be caused by regular link quality fluctuations or node movement.

3.3.2 *Securing Route Discovery*

We now review some protocols that attempt to secure the route discovery process of Source Routing, Distance Vector Routing and Link State Routing.

3.3.2.1 *Securing Route Discovery for Source Routing Protocols*

SRP The Secure Routing Protocol (SRP) by Papadimitratos and Haas [77] is an extension for Source Routing protocols such as DSR. SRP relies on Message Authentication Codes that are checked at the end-points to assure the correctness of topology information retrieved by the route discovery protocol. It assumes existing SAs for every source-destination pair. The use of symmetric cryptography is computationally efficient: Message Authentication Codes [61] are calculated once for every RREQ and RREP; and verified (again only once) at the destination and source, respectively. Relaying nodes only check the protocol format and the forwarding list. The authors claim that their scheme is secure against multiple *non-colluding* malicious nodes. Colluding nodes can successfully mount a tunneling attack.

ARIADNE Ariadne by Hu et al. [45] is an approach towards more strongly securing DSR. The main goal is the identification of malicious nodes on routes and, thus, the ability to route around them in subsequent protocol runs. It uses message authentication codes to achieve end-to-end authentication of RREQs and RREPs. In addition, Ariadne provides authentication for nodes on the path (using keyed hash chains, digital signatures or message authentication codes) to prevent adversaries from removing valid nodes from the returned source route. This does not provide resilience against wormhole attacks. In contrast to SRP, Ariadne requires SAs between all nodes on the path including source and destination which can be a scalability problem.

Ács et al. propose a provably secure routing protocol called endairA [20] based on Ariadne.

3.3.2.2 *Securing Route Discovery for* Reactive *Distance Vector Routing Protocols*

SAODV Zapata and Asokan propose a security extension to AODV [107]: Their SAODV protocol provides authentication of RREQ and RREP packets using digital signatures and lightweight integrity protection of the mutable hop count field with hash chains. Thus, it achieves security against message spoofing. Protection of the hop count is limited since attackers can forward messages without increasing the field. As a result, returned routes might appear shorter than they really are. The maximal possible length reduction depends on the number of malicious nodes on the path. Two colluding nodes can also mount a tunneling attack to reduce the returned path length even further.

ARAN Authenticated Routing for Ad hoc Networks (ARAN) by Sanzgiri et al. [97] is an example for a secure DVR protocol. It secures the traversals of RREQ and RREP packets using public-key cryptography. A source node signs a RREQ with its own private key. At each hop, the signature of the preceding node is verified and the own signature is appended to the packet. On the one hand, this approach gives an attacker no chance to alter the packet content, but it is quite expensive: Public-key cryptography is used at *every* hop which places a heavy computational burden on relaying nodes.

Instead of using hop count as routing metric, ARAN uses timestamps which prevent successful tunneling attacks; attackers can still establish a tunnel but cannot achieve faster delivery times, i. e., create "shorter" routes. ARAN's timestamps do not protect against wormhole attacks.

CASTOR In their 2010 paper, Galuba et al. proposed Continuously Adapting Secure Topology-Oblivious Routing (Castor) [31, 30]. It is a reactive routing protocol but differs from other protocols in this class in various ways. It is based on the following core concepts:

Implicit Route Discovery. Castor does not have an explicit route discovery phase. Instead, Data Packets (PKTs) are flooded through the network if no route is known to the destination. The routes are then built using Acknowledgments (ACKs) that are replied by the destination upon PKT reception.

Reliability as Distance Metric. Castor attempts to find the most reliable route, tackling accidental as well as deliberate packet loss. Reliability is defined as the past behavior, i. e., packet delivery rate for a single neighbor.

The authors show that Castor is resilient against all attacks presented in Section 3.3.1 while requiring only end-to-end SAs. We discuss Castor in more detail in Section 4.2.

3.3.2.3 *Securing Route Discovery for* Proactive *Distance Vector Routing Protocols*

Hu et al. have proposed Secure Efficient Ad hoc Distance vector (SEAD) protocol [42], which is based on DSDV. SEAD secures the sequence numbers and metric fields of DSDV route state messages, thus, preventing non-colluding malicious nodes from decreasing the advertised distance to other nodes in the network. This prevents adversaries from attracting traffic because they cannot forge *better*, i. e., shorter, routes. Security is achieved using efficient one-way hash chains and Merkle hash trees, obviating the need for expensive public-key cryptography.

3.3.2.4 *Securing Route Discovery for Link State Routing Protocols*

The Secure Link State Routing Protocol (SLSP) [78] was proposed by Papadimitratos and Haas and secures the exchange of LSUs within a certain hop count radius around the source. Similarly to SEAD, hash chains are used to secure the hop count field. LSUs are signed using the nodes' private keys. The corresponding public keys are distributed in separate packets which are broadcast regularly to a node's neighborhood (within the specified hop count radius). This way, nodes only need to validate a limited amount of signatures and store a limited amount of public keys.

3.3.3 *Securing Data Transmission*

Castor, as we have introduced in Section 3.3.2.2, does not use an explicit route discovery phase but rather combines secure route discovery and data transmission. It uses acknowledgments and reliability metrics to correlate failures with specific routes. Two precursory protocols to Castor exclusively deal with the issue of secure data transmission.

SMT AND SSP Papadimitratos and Haas have developed the Secure Message Transmission (SMT) and Secure Single-Path (SSP) protocols [79, 80]. SMT is a multipath routing protocol that is largely independent of the underlying route discovery mechanism. One requirement is that it returns multiple routes to a destination. SMT disperses messages using erasure coding: m out of n pieces suffice to reconstruct the message at the destination. The n pieces are transmitted over node-disjoint paths to achieve robustness against (adversary induced) losses and avoid the need for retransmissions if at least m pieces are received at the destination. SMT uses message authentication codes to validate the integrity and authenticity of the individual pieces at the destination. The destination then provides positive feedback for the received pieces. This feedback is used for 1) determining the paths to be used and 2) dynamically adjusting the parameters m and n.

SSP can be seen as a SMT configuration with fixed $m = n = 1$. This relaxes the requirements for the route discovery scheme because it only needs to discover a single route. Due to the redundancy and lower delay (fewer retransmissions), SMT is better suited for applications that require real-time communication. On the other hand, SSP exhibits less overhead while still operating securely.

3.3.4 *Discussion*

Here, we discuss the security properties of the protocols presented above. In Table 3.1, we give a graphical overview of the protocols and their resistance against the various attacks. Resistance is rated from very strong (■) to completely vulnerable (■). Attacks that have no effect on a certain protocol are marked with (■).

Most protocols, i. e., SRP, Ariadne, SAODV, SEAD, ARAN and SLSP, secure the route discovery process with a focus on the spoofing attack. Blackhole and greyhole attacks on route discovery do not have a severe impact on these protocols as an adversary basically removes itself from the view of other nodes (■, ■). However, they cannot thwart the same attacks on data transmission. They *can* provide secure data transmission in conjunction with SMT or SSP. This requires interaction between both protocols: Consider the situation where an adversary always plays along during route discovery but mounts a blackhole attack against data traffic. SMT/SSP will detect the misbehaving node(s) and trigger the route discovery mechanism. If the route discovery protocol does not receive any information about the misbehaving nodes from SMT/SSP, the same routes will be returned because the adversary will, again, adhere to the protocol.

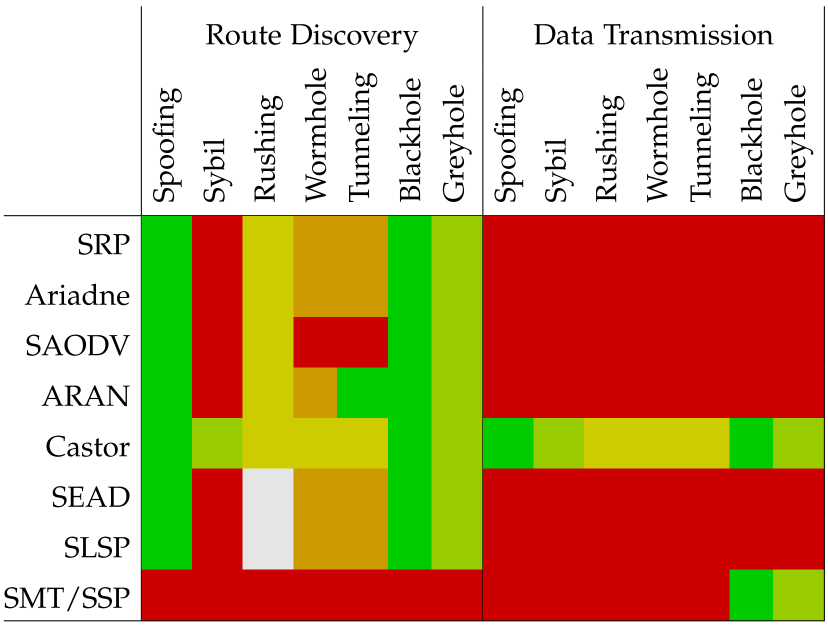


Table 3.1: Resistance of unicast routing protocols against various attacks.

Sybil attack resilience is an issue of trust. If an adversary has been able to compromise the cryptographic material (private keys) of multiple nodes, it can consequently operate under multiple identities. All protocols presented here provide no means of Sybil *detection* and are, thus, unable to validate if a one-to-one mapping of entity to identity exists. Detection schemes based on distributed packet monitoring have been proposed, e. g., [101], but they can only provide *reactive security*. In addition, such schemes introduce overhead for exchanging monitoring messages and a non-negligible probability of false positives.

All reactive protocols are vulnerable to the rushing attack as long as they deploy the duplicate suppression mechanism for flooding. A counter-measure introduced by Hu et al. [44] is based on randomized RREQ forwarding and secure route delegation in which neighboring nodes verify that a node is a legitimate forwarder (■). Rushing has no effect on proactive protocols because no RREQs are transmitted that can be "rushed" to the intended receiver. Consequently, SEAD and SLSP are immune to this type of attack (■).

We note that all route discovery protocols can—in theory—be secured against wormhole attacks with packet leashes [43]. However, the use of packet leashes imposes new system requirements: *Geographical* leashes require each node to know its own geolocation and assume loosely synchronized clocks among all nodes. *Temporal* leashes, on the other hand, require tightly synchronized clocks, which could be achieved using protocols such as Network Time Protocol (NTP) assuming the availability of an appropriate server. The problem is that both methods introduce new attack surfaces [81, 9]. A simple DoS attack on GPS and NTP could jeopardize the scheme (■). Packet leashes cannot be used for preventing tunneling attacks. However, ARAN is able to defend against tunneling without modification due to the use of timestamps in its RREQ packets.

Castor is a peculiar protocol in the way it secures route discovery and data transmission. Instead of deploying mechanisms to explicitly detect specific attacks such as wormholes or tunnels, Castor uses the (possibly high) capacity of adversary links opportunistically. As long as ACKs are received over the reverse path, data packets continue to be transmitted over the corresponding links. As soon as the adversary starts to drop packets, however, the reliability for this node decreases until either another reliable neighbor is chosen or the protocol falls back to flooding. Similarly, the ratings of the multiple identities of a Sybil adversary will gradually decrease when it starts dropping packets. In summary, Castor is vulnerable to "traffic attracting" attacks such as rushing and wormholes per se (■). However, routing is only negatively affected when combined with, e. g., a blackhole attack. In this case, Castor is able to recover quickly if an alternative path exists. This property makes Castor the only standalone protocol reviewed here that—by design—withstands most of the presented attacks.

3.4 SECURITY OF MANET MULTICAST ROUTING

While a lot of work has been carried out in the area of secure *unicast* routing, the literature on secure multicast in MANETs is sparse.

A significant portion of the available work in this area has considered the issue of group key management, e. g., [92]. While key management, especially for multicast, is an important and complex issue, we reiterate that we are not concerned with key management in this thesis. Instead, we focus on securing the routing mechanism.

3.4.1 *Attacks on Multicast*

Nguyen and Nguyen have identified fundamental attacks on MANET multicast routing [74]. Their findings include attacks such as rushing, blackholes and tunneling which we have discussed in Section 3.3.1.

3.4.2 *Securing Multicast*

We identify two basic operations for multicast routing: *1*) group discovery and maintenance, and *2*) data transmission. These operations are similar to route discovery and data transmission for unicast routing.

3.4.2.1 *Securing Group Discovery and Maintenance*

S-MAODV Roy et al. provide an extensive security analysis of the Multicast Ad Hoc On-Demand Distance Vector (MAODV) routing protocol [93]. They identify attacks on MAODV, which are specific to tree-based multicast: In particular, the authors describe attacks on tree pruning, link repair and the partition merge process. They all result in some kind of DoS such as preventing nodes from joining the multicast group or forcefully partitioning the multicast tree.

The authors propose an authentication framework for MAODV which is based on neighbor authentication, group leader authentication, tree-key dissemination for group members and hop count authentication. We will refer to this secured version of MAODV as S-MAODV.

BSMR Byzantine-Resilient Secure Multicast Routing (BSMR) is a secure on-demand tree-based multicast protocol [21]. The authors use an authentication framework, protecting multicast groups from external adversaries, similar to S-MAODV. In BSMR, RREQs and RREPs are always flooded so the protocol is able to find an adversary-free path if one exists. BSMR introduces MRATE messages, which contain a source's current transmission rate, and which are flooded periodically to all multicast members. Using the advertised transmission rate and the locally perceived rate of incoming packets, nodes are able to detect data transmission attacks of upstream tree members. This way, BSMR remains resilient even against blackhole attacks.

3.4.3 *Discussion*

Security in MANET multicast routing is clearly an understudied research area. We identified two papers that consider this topic. Both publications address the problem of securing tree-based multicast [93, 21]. Work on secure mesh-based and stateless multicast seems to be missing.

Curtmola and Nita-Rotaru [21] provide simulation results on the resistance of S-MAODV and BSMR against various attacks. Both protocols withstand spoofing attacks due to their authentication frameworks. BSMR withstands rushing, tunneling[2] and blackhole attacks (■), while S-MAODV's performance drops significantly even with

2 The authors use the term "wormhole attack" to describe an attack we termed as tunneling ("The adversaries use the lowcost *appearance* of the wormhole [...]")

	Spoofing	Sybil	Rushing	Wormhole	Tunneling	Blackhole	Greyhole
S-MAODV		?					
BSMR		?					

Table 3.2: Resistance of multicast routing protocols against various attacks.

		Group Discovery and Maintenance	Data Transmission
Stateful	Tree-based		
	Mesh-based		
Stateless			

Table 3.3: Addressed issues of secure MANET multicast in the literature.

a small fraction of malicious nodes present in the network (■). However, BSMR incurs more overhead: 40–100% more than S-MAODV due to the flooding of RREQ, RREP and MRATE packets, depending on the attack scenario. The authors do not provide empiric proof of how their protocols handle Sybil, tunneling and greyhole attacks. We do not predict the resistance against Sybil attacks (marked with '?'). We expect resistance against wormhole attacks to be similar to that against tunneling since both attacks target the protocols' hop counter. Greyhole resistance is expected to be weaker than blackhole resistance since greyholes are more difficult to detect.

With BSMR, a full-fledged solution for secure tree-based multicast was proposed (■), but work on secure mesh-based and stateless protocols is missing (□; Table 3.3). It seems contradictory that even though mesh-based and stateless multicast schemes appear to be the go-to candidates for MANETs in terms of scalability (Section 2.5), security-related work has not been carried out for these types of protocols.

Part II

XCASTOR: A SCALABLE AND SECURE EXPLICIT MULTICAST ROUTING PROTOCOL

4 DESIGN

This chapter addresses the core problem of the thesis:

The design of a secure and scalable routing protocol for MANETs.

We discuss why an existing secure unicast protocol presents itself as a substrate for a scalable and secure multicast extension. We develop such an extension and describe the design process and the choices that we made.

4.1 CHOOSING THE SUBSTRATE

Currently, no multicast routing protocol for MANETs exists that is both scalable and secure. As a result, we can either extend existing solutions with the desired properties or create a new protocol from scratch. In particular, we consider the following options:

1. Choose a *scalable multicast* routing protocol and make it *secure*; or
2. choose a *secure multicast* routing protocol and make it *scale*; or
3. choose a *secure and scalable unicast* routing protocol and add *multicast* support[1]; or
4. design a *secure and scalable multicast* protocol from scratch.

We consider *security by design* as a fundamental approach towards designing our multicast protocol. Castor, as an example for security-by-design unicast routing, was shown to be very robust even against strong attacks. In contrast, mechanisms that retrospectively *add* robustness against attacks can introduce new unexpected attack surfaces (Section 3.3.4). By this, we exclude option 1 (securing a previously unsecured multicast protocol) from further investigation.

In Section 3.4, we have seen that previous work on secure multicast is sparse. Only tree-based multicast has been considered in literature. However, tree-based approaches inherit some drawbacks [56]: *1*) They tend to be unreliable, especially in volatile environments, since only a single path to each destination is known; and *2*) coping with node mobility requires tree maintenance mechanisms, which are frequently triggered under high node mobility. Since tree-based protocols are less suitable for dynamic environments, and due to the lack of secure mesh-based or stateless multicast protocols, we disregard option 2.

This leaves us with options 3 and 4. Since security in routing protocols is not trivially achieved (Section 3.3), we decide against an approach requiring a from-scratch design. We favor to build upon a secure unicast protocol instead. We have identified Castor as the only protocol providing comprehensive security properties among all secure unicast routing protocols. For this reason, we choose Castor to serve as a substrate for achieving our goal of a secure and scalable multicast routing protocol.

1 We do not imply that scalable multicast automatically follows from scalable unicast. Retaining scalability requires a careful design of the multicast extension.

4.2 CASTOR IN DETAIL

A rough introduction of Castor was given in Section 3.3.2.2. It is necessary to understand Castor's design in detail before diving into solutions for multicast extensions. What follows is a short summary of the protocol details [31].

4.2.1 *Packet Format*

Castor does not have an explicit route discovery phase. Instead, when attempting to communicate with a destination, user data is directly distributed within Data Packets (PKTs). Acknowledgments (ACKs) are returned by the destination and used as feedback.

PKT. The Data Packet is a tuple $pkt = \langle s, d, H, b_k, f_k, e_k, \mathcal{P}\rangle$: s and d are the source and destination identifiers, respectively; H is the flow identifier; b_k is the PKT identifier; f_k is the flow authenticator; e_k is the PKT authenticator; and $\mathcal{P}$ is the user payload, which may be encrypted and must be integrity-protected. The index k denotes the kth PKT of flow H.

ACK. The Acknowledgment consists of only one field: $ack = \langle a_k\rangle$: a_k is the unencrypted version of e_k.

4.2.2 *Cryptographic Mechanisms*

Castor maintains correct routing state by a scheme that satisfies two properties: *1*) Only the source node is able to generate packet ids b_ks belonging to flow H. *2*) Valid ACKs can only be received by intermediate nodes if the destination has actually received the corresponding PKT. The here scheme discussed here is based on Merkle hash trees. We denote $\textsc{Enc}_{K_{sd}}(\cdot)$ and $\textsc{Dec}_{K_{sd}}(\cdot)$ as a pair of symmetric encryption and decryption functions; K_{sd} is a shared key between s and d. $H(\cdot)$ is a cryptographic hash function, i. e., it is hard to compute preimages.

PKT GENERATION If s wants to communicate with d, s pre-computes a "flow": *1*) generated random nonces $\langle a_1, \dots, a_w\rangle$ act as ACK authenticators, where w is the number of PKTs that can be sent with this flow; *2*) the PKT identifiers are calculated as $b_k = H(a_k)$; *3*) a Merkle hash tree with $\langle H(b_1), \dots, H(b_w)\rangle$ as leaves is built, where its root becomes the flow identifier H.

For every PKT, s *1*) sets $f_k = \langle x_1, \dots, x_{\log_2 w}\rangle$ (siblings on the path from leaf $H(b_k)$ to the root H), and *2*) computes $e_k = \textsc{Enc}_{K_{sd}}(a_k)$.

FLOW VERIFICATION Upon reception of a new PKT, each node verifies that the packet identifier b_k actually belongs to flow H by calculating[2]

$$H\Big(\dots H(H(H(b_1)\,\|x_1)\,\|x_2)\,\|\dots x_{\log_2 w}\Big) \stackrel{!}{=} H. \tag{4.1}$$

2 Equation 4.1 is only an example illustrating the calculation of the tree's root starting from b_1. For other b_k with $k > 1$ the concatenation order depends on the position of b_k in the Merkle hash tree, i. e., whether x_i is a sibling to the left or right.

If the verification fails, the PKT is dropped. Otherwise, b_k is stored and the PKT is forwarded.

PKT VERIFICATION The destination d performs source authentication in addition to flow verification:

$$H(\text{Dec}_{K_{sd}}(e_k)) \stackrel{!}{=} b_k. \tag{4.2}$$

If the verification succeeds, the destination generates an ACK with $a_k = \text{Dec}_{K_{sd}}(e_k)$.

ACK VERIFICATION Nodes receiving an ACK compute $H(a_k)$ and check whether it belongs to any previously seen b_k. If this is the case, the routing state is updated, a_k is stored, and the ACK is rebroadcast. Otherwise, the ACK is discarded.

4.2.3 *Forwarding*

Each node maintains a reliability estimator $s_{H,j} \in [0,1]$ for every encountered flow H and every neighbor h_j. When receiving a PKT, the forwarding node performs flow verification and, if successful, determines the most reliable node $\hat{h}$ according to the reliability estimator with $p = \max_j s_{H,j}$. A node will broadcast a PKT with a probability of $e^{-\gamma p}$ to all neighbors or unicast to $\hat{h}$ with probability $1 - e^{-\gamma p}$. The parameter $\gamma > 0$ controls the bandwidth investment for discovering new routes. Note that initially $p = 0$, and thus a PKT is always broadcast. Upon forwarding, a timer T_{b_k} is started that times out after T_{ACK}.

HANDLING OF DUPLICATES Duplicate PKTs, i. e., PKTs containing a previously seen triple $\langle b_k, e_k, \mathcal{P}\rangle$, will be dropped. If b_k has been seen before but either or both other fields are new, the PKT needs to be forwarded because intermediate nodes cannot verify the integrity and authenticity of e_k and $\mathcal{P}$, that is, they cannot differentiate between legitimate and illegitimate e_ks and $\mathcal{P}$s. Duplicate ACKs are always dropped.

If a node receives a duplicate PKT from a new neighbor and if a valid ACK has already been received, the ACK will be retransmitted to that neighbor.

4.2.4 *Reliability Estimators*

Castor's reliability estimator $s_{H,j}$ is an arithmetic average of two parameters, $s^a_{H,j}$ and $s^f_{H,j}$. The two values are exponential averages of packet delivery rates. Let $\alpha^a_{H,j}$ and $\beta^a_{H,j}$ be running averages of successful and failed deliveries, respectively. Then $s^a_{H,j} = \frac{\alpha^a_{H,j}}{\alpha^a_{H,j}+\beta^a_{H,j}}$. $s^a_{H,j}$ is *decreased* as

$$\alpha^a_{H,j} = \delta\alpha^a_{H,j},$$
$$\beta^a_{H,j} = \delta\beta^a_{H,j} + 1,$$

and *increased* as

$$\alpha^a_{H,j} = \delta\alpha^a_{H,j} + 1,$$
$$\beta^a_{H,j} = \delta\beta^a_{H,j},$$

with $0 < \delta < 1$ defining how fast the values will change. $s^f_{H,j}$ is updated analogously.

UPDATING Two events can trigger an update in the reliability estimators:

1) A nodes receives a valid ACK (a_k) from a neighbor h_j before T_{b_k} times out: If the PKT (b_k) has been broadcast by the node and a_k was not (!) the first ACK received for b_k, only $s^a_{H,j}$ is increased. Otherwise, both $s^a_{H,j}$ and $s^f_{H,j}$ are increased. The superscripts a and f refer to an update on "all" or "first" ACKs, respectively.

2) T_{b_k} times out before a valid ACK was received: If the PKT was broadcast, no estimator changes. Otherwise, both $s^a_{H,j}$ and $s^f_{H,j}$ are decreased.

By using the second estimator $s^f_{H,j}$, Castor gives preference to low-latency routes that typically consume less bandwidth.

4.3 EXTENDING CASTOR WITH MULTICAST SUPPORT

We aim towards extending Castor with multicast support while maintaining its simplicity and security features. Explicit Multicast (Xcast) appears to be a suitable approach for Castor since it allows sender-side verification of ACKs, which is necessary for selective PKT retransmissions.

In the following sections, we present the design process and discuss various alternatives. The final design of our *Xcastor* protocol is summarized in Section 4.4.

4.3.1 *A First Approach using Xcast*

We apply the Xcast concept on Castor routing. First of all, we need to extend the Castor header so that it can accommodate multiple destinations $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$ ($n = |\mathcal{D}|$). The source will include all multicast group members in the header of each multicast packet.

Upon reception, intermediate nodes need to decide how to further process the packet. In Castor, nodes can choose to either unicast or broadcast. We generalize this approach to cope with PKTs addressed a set of destinations $\mathcal{D}$ ($\text{pkt}(\mathcal{D})$): We let Castor decide whether to unicast or broadcast $\text{pkt}(\mathcal{D})$ for each $d \in \mathcal{D}$. But instead of transmitting n individual packets, i. e., one for every d, we transmit messages with the same next hop $h_i \in \mathcal{N}$ (with $\mathcal{N}$ being the neighbor set) as a single packet (we denote all destinations with the same next hop h_i as *forwarder set* $\mathcal{F}_i$, with $i = 1, \ldots, |\mathcal{N}|$). Similarly, for all destinations that Castor chooses to broadcast to (denoted as the broadcast set $\mathcal{B}$), we perform a single broadcast including all $d \in \mathcal{B}$ in the header. The pseudocode of this scheme can be found in Algorithm 1. The checks for empty set $\emptyset$ are included to avoid transmissions with no receivers.

In the case that Castor selects a single neighbor as the next hop for all destinations, the protocol issues a single unicast transmission to that neighbor. Similarly, in the case that no reliable routes to any destination are known, a single broadcast message is transmitted. In any other case, more than one transmission takes place.

4.3.2 *Packet Merging*

In Algorithm 1, a node h_i might receive two packets: a unicast transmission including the forwarder set $\mathcal{F}_i$ as well as a broadcast transmission including the (disjoint) broad-

Algorithm 1 Straightforward Xcast on Castor

```
function FORWARD_MULTICAST_PACKET(pkt(𝒟))
    for all ℱ_i ← {d ∈ 𝒟 : NEXT_HOP(d) = h_i} do
        if ℱ_i ≠ ∅ then
            UNICAST pkt(ℱ_i) to h_i
        end if
    end for
    ℬ ← {d ∈ 𝒟 : d ∉ ⋃ℱ_i}
    if ℬ ≠ ∅ then
        BROADCAST pkt(ℬ) (to 𝒩)
    end if
end function
```

cast set $\mathcal{B}$ ($\mathcal{F}_i \cap \mathcal{B} = \emptyset$). In that case, we need to make sure that every neighbor has received both packets before further processing so that it learns all destinations it is supposed to forward pkt to. There are several options to achieve this:

1. Upon receiving a UNICAST PKT, wait for a certain delay Δt for a potential subsequent BROADCAST transmission. If the latter is received, the destinations of both packets have to be merged before passing $\mathrm{pkt}(\mathcal{F}_i \cup \mathcal{B})$ to FORWARD_MULTICAST_PACKET. Upon receiving a BROADCAST PKT, a node is not required to wait since BROADCAST is (in the algorithm) always preceded by UNICAST. We assume that packet reception adheres to the sending order.

2. Unite the broadcast set $\mathcal{B}$ and $\mathcal{F}_i$ so that $\mathcal{F}_i' = \mathcal{F}_i \cup \mathcal{B}$ and include it in unicast PKTs. This avoids the need of introducing a reception delay: the first PKT already contains all relevant destinations for h_i, and thus a subsequent broadcast can be ignored.

3. Transmit each packet exactly once by including a mapping for all $h_i \to \mathcal{F}_i$ as well as for $\mathcal{N} \to \mathcal{B}$ in the header so that the payload is transmitted only once. Using these mappings, each node needs to check its forwarding responsibility: Node h_i is responsible for destinations $\mathcal{F}_i \cup \mathcal{B}$. If $\mathcal{F}_i = \emptyset$ and $\mathcal{B} = \emptyset$, the PKT is dropped.

We favor option 3: it is maximally efficient w. r. t. data transmission overhead since the payload is only transmitted once at each forwarding node.

4.3.3 *Group Keys: Header Size Revisited*

The Xcast header size problem is amplified in Algorithm 1: Since Castor uses PKT authenticators $e_k = \mathrm{Enc}_{K_{sd_i}}(a_k)$, each PKT needs to include an encrypted version of a_k for every member d_i of the multicast group. More formally, we change the original e_k to a list

$$e_k' = \left\langle \mathrm{Enc}_{K_{sd_1}}(a_k), \ldots, \mathrm{Enc}_{K_{sd_i}}(a_k), \ldots, \mathrm{Enc}_{K_{sd_n}}(a_k) \right\rangle .$$

Assuming a ciphertext size of $|\mathrm{Enc}_{K_{sd_i}}(a_k)| = 32$ bytes $= 256$ bits and a group size of $n = 8$ members, this yields a total overhead of 32 bytes $\times\, n = 256$ bytes—just for the PKT authenticators. This is already ~17 % of the maximally allowed Ethernet frame payload (1500 bytes) [47]. We present a solution that reduces this overhead:

GROUP KEYS Group keys [3] are symmetric keys that are shared with every member of a group. The ACK authenticator is encrypted with a group key K_g and, thus only needs to be included once. We set

$$e''_k = \text{Enc}_{K_g}(a_k)\,. \tag{4.3}$$

This approach allows the header to retain its size except for the additional destination addresses. Note that K_{sd_i}s are required in any case as the source needs to distribute the group key over a secure channel prior to the multicast communication.

With group keys, *insider* adversaries[3] become a problem [6]: They can easily impersonate any other group node and forge messages, e. g., such that they appear to originate from the legitimate multicast source[4].

However, if we assume a trust relationship between all group members, then this is no longer an issue. For now, we accept group keys as a reasonable approach to retain a compact header size (we will later address insider adversaries again).

Note that we are not concerned here with issues such as group access control, initial distribution of the group key, or re-keying in case of group membership changes. These are all relevant issues, but beyond the scope of this thesis.

4.3.4 *ACK Authentication Problem*

In the current scheme, all group nodes would create ACKs with the same a_k, making them indistinguishable. We could fix this by including d_i as second ACK field. This allows for the following attack.

ACK ORIGINATOR ATTACK We must assume that the adversary knows about all or part of the multicast group (which he can easily determine from any multicast PKT header since we use *explicit* multicast). In addition, let us assume that an adversary receives an ACK from *any* legitimate multicast destination d_i. Since all ACKs for one PKT have the same a_k, the adversary could fake and send ACKs appearing to originate from any other group member $d_j \neq d_i$ by simply changing the unsecured source address field from d_i to d_j. This will deceive intermediate relay nodes as well as the source node into believing that the PKT was properly received by d_j: The adversary is able to cut off all but one destination (d_i) from the multicast group without the source noticing (DoS).

The presented attack exploits the fact that intermediate nodes cannot authenticate the originator of the message. They can only infer that at least one multicast destination must have received the PKT if the received a_k is valid. Thus, we need to alter the ACK authentication mechanism.

Signed ACKs. One option that comes to mind is the use of signed ACKs: The receiver signs each ACK with its private key. Intermediate nodes could then verify the

3 In contrast to the *internal* adversary (Section 3.2.3), who actively takes part in the routing protocol, we define the *insider* adversary to be also part of the multicast group and in possession of the shared group key.

4 Note that for unicast communication, this would not pose a problem: Both parties (source and destination) know which messages originated from themselves. So, an unknown message authenticated with the shared key must have originated from the other party (always assuming that neither node was compromised).

origin of any received ACK if they know the public key of the ACK source. This requires every node to retrieve the public key of any other node in the network[5]. Castor's design is based on weaker security assumptions, i. e., the existence of end-to-end SAs, only. We prefer a solution that maintains the original design assumptions and relies on light-weight symmetric-key cryptography.

Individual Flows. Using independent per-destination flows solves the ACK identification problem. In essence, this approach includes multiple independent Castor headers in a single PKT. The only advantage of this approach compared to *multicast via unicast* is that the payload is transmitted only once. While this might seem to be a feasible solution, per-destination headers create a significant overhead: Individual flows would require inclusion of multiple flow IDs and flow authenticators in the PKT header.

Individual PKT identifiers. A more efficient solution is the following: Using K_{sd_i}s, we introduce *individual* PKT identifiers $b_{k,i}$ for every d_i as

$$b_{k,i} = H\left(\text{Enc}_{K_{sd_i}}(a_k)\right). \tag{4.4}$$

These individually encrypted and hashed versions of a_k are appended to the original b_k:

$$b'_k = \langle b_k, b_{k,1}, \dots, b_{k,i}, \dots, b_{k,n} \rangle. \tag{4.5}$$

The original b_k needs to remain in the header since it is used for forward flow verification, i. e., to validate that this PKT belongs to the indicated flow.

Upon PKT reception, a destination d_i calculates $a_k = \text{Dec}_{K_g}(e''_k)$ (Equation 4.3), and returns an ACK containing

$$a'_{k,i} = \text{Enc}_{K_{sd_i}}(a_k). \tag{4.6}$$

Upon reception of a'_k, each forwarding node can verify that the originator is indeed a *specific*, legitimate group member, i. e., check whether $H(a'_{k,i})$ belongs a corresponding $b_{k,i}$.

We note that the order of b'_k does not need to be protected. Consider the following attack:

REORDERING ATTACK Let an adversary swap $b_{k,i}$ and $b_{k,j}$ in a forwarded PKT, then the returned $a'_{k,i}$ from d_i does not match the expected $b_{k,j}$. The ACK is consequently discarded.

The individual PKT identifiers offer several advantages: *1*) PKT header size is increased only by the size of $b_{k,i}$ per multicast receiver, i. e., a total additional size of $n \times |H(\cdot)|$ compared to a standard Castor PKT; the ACK retains its size; *2*) The insider adversary problem is solved as a side effect since $a'_{k,i}$ cannot be forged by other group members.

5 One method to achieve this is as follows: The destination's public key is signed by the source and included in the PKT. Then, the signed ACK can be verified by any node that previously forwarded the PKT.

4.3.5 *Optimizing PKT Size*

If we take a closer look at the current PKT format, we notice two things:

1. e''_k from Equation 4.3 does not need to be encrypted: An adversary is only able to forge valid ACKs if it has access to both a_k *and* K_{sd_i}. Consequently, we can remove the encryption of e''_k such that the PKT authenticator is set to
$$e'''_k = a_k. \tag{4.7}$$
As a result, we no longer require a group key K_g for securing the header. However, we still need it to provide data integrity protection or encryption.

2. Since a_k is now transmitted in plaintext and $b_k = H(a_k)$ which can be computed locally by every node, we can remove b_k from the header to save bandwidth:
$$b''_k = \langle b_{k,1}, \ldots, b_{k,i}, \ldots, b_{k,n} \rangle . \tag{4.8}$$

4.4 SUMMARY: XCASTOR

We have presented the design process including various alternatives in the previous sections. Here, we want to summarize the final design. We call it *Xcastor*.

4.4.1 *Packet Format*

The packet format for Xcastor looks as follows:

$$\begin{aligned} \mathrm{pkt} = \Big\langle s, &\overbrace{\langle h_1, \mathcal{F}_1 \rangle, \ldots, \langle h_j, \mathcal{F}_j \rangle, \ldots, \langle h_m, \mathcal{F}_m \rangle}^{\text{forwarder mapping}}, \mathcal{B}, \\ &H, \langle b_{k,1}, \ldots, b_{k,i}, \ldots, b_{k,n} \rangle, f_k, a_k, \mathcal{P} \Big\rangle, \end{aligned} \tag{4.9}$$
$$\mathrm{ack} = \langle e_{k,i} \rangle , \tag{4.10}$$

with

H, f_k as the flow identifier and authenticator as in Castor,
$e_{k,i} = \mathrm{ENC}_{K_{sd_i}}(a_k)$ as the ACK authenticator,
$b_{k,i} = H(e_{k,i})$ as the individual PKT identifiers, and
$\bigcup_{j=1}^{m} \mathcal{F}_j \cup \mathcal{B} \subseteq \bigcup_{i=1}^{n} d_i$ containing the destinations.

4.4.2 *Packet Processing*

When a node j receives a PKT, it checks whether it is included in the forwarder list. If not, i. e., if $\mathcal{F}_j \cup \mathcal{B} = \emptyset$, the PKT is discarded. Otherwise, j removes all $\mathcal{F}_i$ for $i \neq j$ and the corresponding $b_{k,i}$ values and then continues processing.

What follows, is duplicate checking: $b_{k,i}$s previously encountered are removed from pkt and a matching ACK that has previously been received is retransmitted to the sender.

FLOW VERIFICATION Flow verification requires an additional hash operation compared to Castor (Equation 4.1): a_k has to be hashed so that $b_k = H(a_k)$ can be verified.

$$H\Big(\dots H(H(H(H(a_k))\,\|x_1)\,\|x_2)\,\|\dots x_{\log_2 w}\Big) \stackrel{!}{=} H. \tag{4.11}$$

PKT VERIFICATION If the remaining destination set includes j, i. e., the PKT reached a destination, d_j and $b_{k,j}$ are removed from pkt and the pair $\langle b_{k,j}, a_k \rangle$ is verified using Equation 4.12.

$$H\Big(\textsc{Enc}_{K_{sd_j}}(a_k)\Big) \stackrel{!}{=} b_{k,j}. \tag{4.12}$$

If successful, $ack = \langle \textsc{Enc}_{K_{sd_j}}(a_k) \rangle$ is returned to the PKT sender.

PKT FORWARDING The remaining forwarding process is derived from Section 4.2.3. The main difference is that an intermediate node essentially performs the Castor lookup steps for all *subflows* $H_i = \langle H, d_i \rangle$ individually. The concrete changes include:

- Reliability estimators are stored and maintained as $s_{H_i,j}$.
- Hence, route lookup is performed individually for every H_i and the results are stored in the appropriate forwarding sets $\mathcal{F}_1, \dots, \mathcal{F}_m$ and $\mathcal{B}$.
- Individual timers are started for every $b_{k,i}$ forwarded.

ACK VERIFICATION Since $b_{k,i}$s are unique, ACK processing does not need to change. The only difference is that $H(e_{k,i})$ is calculated instead of $H(a_k)$ and looked up in the list of forwarded $b_{k,i}$ values.

4.4.3 *Xcastor Security*

We provide arguments for the security of Xcastor.

Requirement. ack is authentic, i. e.,

(1) each node is able to verify an ack, and
(2) only the destination (and the source) can produce an authentic ack.

Proof. The dependence of $ack = \langle e_{k,i} \rangle$ is as follows:

$$a_k \xmapsto{\textsc{Enc}_{K_{sd_i}}(\cdot)} e_{k,i} \xmapsto{H(\cdot)} b_{k,i}. \tag{4.13}$$

Verification is performed using the one-way[6] hash function

$$H: \quad e_{k,i} \underset{\text{generate (hard)}}{\overset{\text{verify (easy)}}{\rightleftarrows}} b_{k,i}. \tag{4.14}$$

The one-way property directly satisfies (1): Each node can easily verify $e_{k,i}$ by knowing $b_{k,i}$ and H. (2) is satisfied by: (a) the one-way property of H, i. e.,

6 Property of a one-way function: It is easy to compute on every input, but hard to invert, i. e., to find a preimage.

generating the preimage of $b_{k,i}$ is hard for every node; (b) only the destination (and the source) can generate $e_{k,i}$ from a_k (knowledge of shared secret K_{sd_i}); (c) $e_{k,i}$ cannot be derived from both a_k and $b_{k,i}$ due to the properties of Enc and H. The argument for (c) is as follows: Since a_k is chosen at random, and assuming that Enc is secure, $e_{k,i}$ is a pseudo-random value[7] that is unknown to an adversary. Then, assuming "reasonable" properties of H, $b_{k,i}$ is pseudo-random as well. Now suppose that an adversary is able to correctly guess $e_{k,i}$ using a_k and $b_{k,i}$ with a non-negligible probability. Then, $b_{k,i}$ is not pseudo-random since the adversary can distinguish it from a truly random value (truly random values cannot be inverted due to the one-way property of H). By this contradiction, $e_{k,i}$ cannot be learned with non-negligible probability. ■

7 Informally, it means that an adversary cannot practically distinguish it from a (truly) random value.

5 IMPLEMENTATION

As already mentioned in Chapter 1, we implemented both, Castor and Xcastor. We were provided a premature version of Castor in Click. We say premature since it exhibited several bugs and was lacking some crucial features such as timeouts, ACK retransmissions, and flow verification. After fixing these shortcomings, we extended it with the components required for Xcastor.

In this chapter, we only describe the final version of Xcastor: We present implementation choices and describe the individual components of our Click router. The general structure of `Elements` (Section 5.1.1) is the same for Castor.

5.1 THE CLICK MODULAR ROUTER

The Click modular router architecture [60] was chosen as the development framework. It offers several advantages compared to a direct implementation in a simulator framework.

Flexibility. Click is based on C++, can be compiled as a kernel module and thus, theoretically, runs on any Linux-driven machine. At the same time, there exists an integration with the ns-3 network simulator which is convenient for protocol evaluation. The same code can be used for either deployment.

Modularity. Since Click routers are based on "`Element`" classes, it is easy to extend existing protocols by new features. We exploit this property to adapt an existing Castor Click implementation.

Facilitated Debugging. The modular design also allows for convenient debugging of a distributed router network. The framework provides `Elements` that dump packets to a *pcap*-compatible trace file which can then be investigated using a sniffer program such as Wireshark[1].

5.1.1 *Click Elements*

Click receives its name from the possibility of "clicking" together a router using only small `Element` classes that provide elementary functionality. `Elements` share a common interface for pushing or pulling packets. The `Element` ports (which are part of the class interface) are connected using a dedicated Click configuration file. Click even supports the creation of more complex composite or compound `Elements`, which are comprised of multiple basic `Elements`.

`Elements` in Click can have multiple input and output ports. This allows, for example, the usage of dedicated ports for invalid packets, which are then discarded, or, using one port for delivering a PKT and the other for pushing out a generated ACK.

1 Wireshark project homepage: `https://www.wireshark.org/`

```
struct CastorXcastPkt {
        /* Fixed length part */
        uint8_t type;           // "PKT"
        uint8_t hashSize;       // Size of a Hash value (in bytes)
        uint8_t nFlow;          // Number of elements in flowAuth
        uint8_t contentType;    // IP packet inside?
        uint16_t length;        // Total length of the header
        uint16_t kPkt;          // The k-th PKT in current flow
        IPAddress source;       // Source and ...
        IPAddress multicastGrp; // ... multicast address
        Hash flowId;            // Flow ID as in Castor
        Hash flowAuth[nFlow];   // Flow authenticator as in Castor
        Hash pktAuth;           // PKT authenticator
        uint8_t nDests;         // # of Xcast destinations
        uint8_t nNextHops;      // # of forwarders
                                // 2 bytes padding for alignment
        /* Variable length part */
        IPAddress dests[nDests];// Individual Xcast destinations
        Hash pid[nDests];       // Individual packet IDs
        IPAddress nextHops[nNextHops];  // Forwarders
        uint8_t map[nextHops];  // Forwarder responsibilities
}

struct CastorXcastAck {
        uint8_t type;     // "ACK"
        uint8_t encSize;  // Size of a Cipher value (in bytes)
        uint16_t length;  // Total length of the header
        Cipher ackAuth;   // ACK authenticator
}
```

Listing 5.1: Xcastor PKT and ACK header in C++ syntax

A Click configuration visualizer can neatly show the flow of packets in the router configurations, which helps when designing more complex systems.

5.2 IMPLEMENTING XCASTOR IN CLICK

We briefly introduce our implementation in Click. We describe the header format as well as the Click configuration of a Xcastor router. The

5.2.1 *Packet Format*

Click uses the `Packet` class for passing packets from `Element` to `Element`, which is essentially a byte buffer that can also hold some meta data ("annotations"). We provide wrapper classes (`CastorXcastPkt` and `CastorXcastAck`), which expose convenience methods for accessing the Xcastor-specific header fields. This approach does not introduce additional overhead since the getter and setter methods only access the `Packet` byte buffer using appropriate offsets. The byte buffers for PKT and ACK are structured as described in Listing 5.1.

In addition to the provided comments in the listings, we would like to point out some implementation considerations.

- Our protocol implementation provides an interface that allows the application to send IP packets to a specific multicast address. Internally, this multicast address is mapped to a list of destinations. Contrary to our design proposal, we include the multicast group address as a PKT header field: the `multicastGrp` field allows the application to identify the multicast group addressed by the PKT.
- The `kPkt` field is needed to determine whether the hash values in `flowAuth` are left or right siblings in the Merkle hash tree, i. e., it is needed for flow authentication.
- The `nextHops` and `map` fields are the result of our forwarding list. Together, they provide a mapping of forwarders to destinations (see Section 4.3.2). `map` defines the range a destination is responsible for. It is implemented as follows:

  `nextHop[0]` is responsible for the destination range `dests[0]` to `dests[map[0]-1]`,
  `nextHop[1]` is forwarder for `dests[map[0]]` to `dests[map[0]+map[1]-1]`,
  `nextHop[2]` for `dests[map[0]+map[1]]` to `dests[map[0]+map[1]+map[2]-1]`, etc.

  This appears to be the most efficient choice w. r. t. header size. Considering the alternative, i. e., map from destination to next hop, the `map` array would have `nDests` fields which would always be equal to or larger than `nNextHops`.
- Since we choose SHA-1 as a hash algorithm and AES-128 in ECB mode for encryption, the actual sizes for `Hash` and `Cipher` are 20 and 32 bytes[2], respectively.
- We adhered to a 4-byte aligned format for performance reasons.
- Some notes on the Xcastor header sizes compared to Castor:
  - The PKT header increases by $20 + 4 = 24$ bytes per destination.
  - Since the types for PKT and ACK authenticators are swapped in Xcastor, ACKs are 12 bytes larger, while 12 bytes are removed from PKTs.

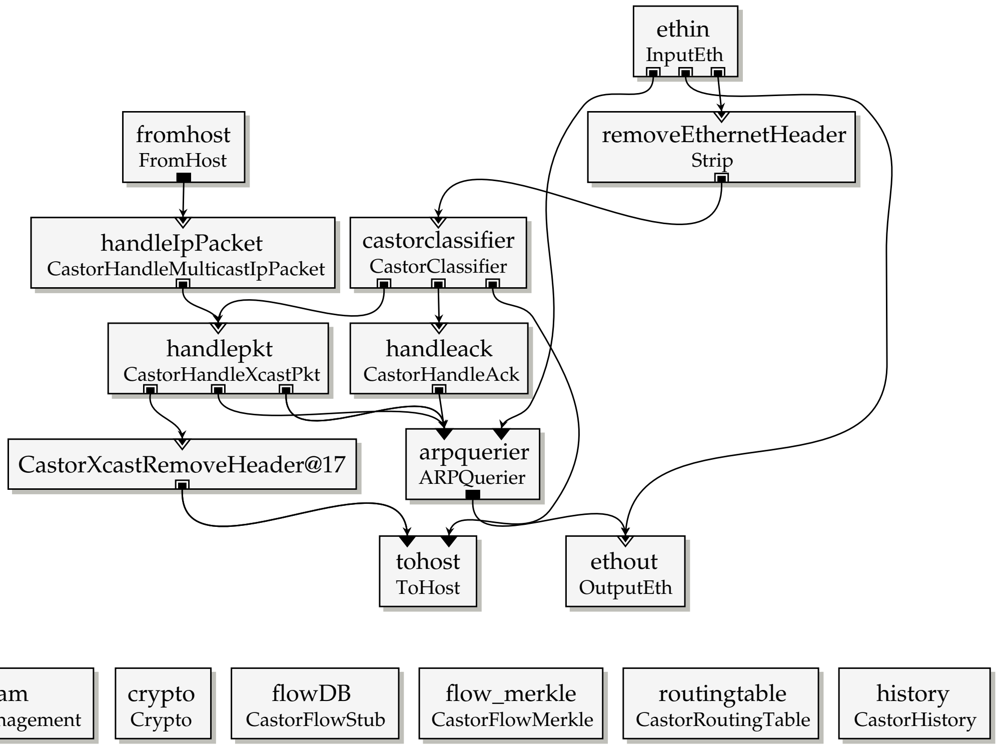
5.2.2 *Elements*

We provide an overview of the Click configuration in Figure 5.1. All composite `Elements` in this figure are presented in more detail in Figure 5.2 to 5.4. The figures were created using the Clicky GUI program which is part of the Click framework and visualize the Click configuration files.

For better coherence, we include the `Element` descriptions in the appropriate captions. A readability note: The first line of each box contains the instance name and the second line the `Element`'s type. A single line ending with `@<NUMBER>` indicates an unnamed `Element` instance of this type.

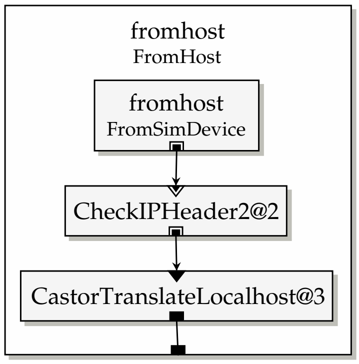
---

2 AES in ECB mode outputs ciphers that are multiples of its 16-byte block size. The smallest multiple of 16 larger than 20 is 32; the hash value needs to be padded.

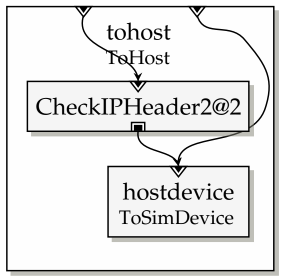


IP packets from the local host are pushed to **handleIpPacket** to prepend the Xcastor header. The following processing (**handlepkt**) is identical to that of other incoming PKTs: Forwarded PKTs and ACKs (**handleack**) are sent to an **arpquerier** which prepends an appropriate Ethernet header and pushes the frames to the transmission queue. The header is removed from PKTs addressed to the local host and pushed to **tohost**. The unconnected **Elements** at the bottom are shared by some of the above **Elements**. For example, the routing table is used in **handlepkt** to look up next hops while it is updated in **handleack**.

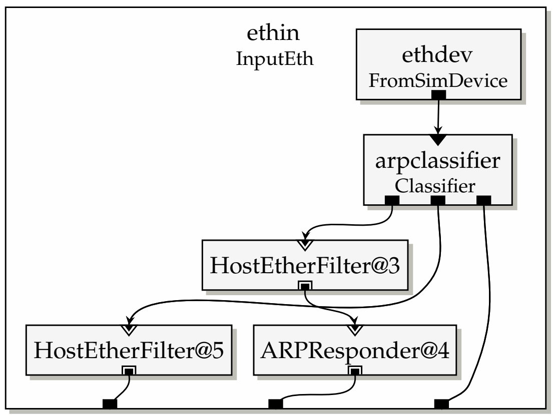
Figure 5.1: Overview of our Xcastor Click implementation.



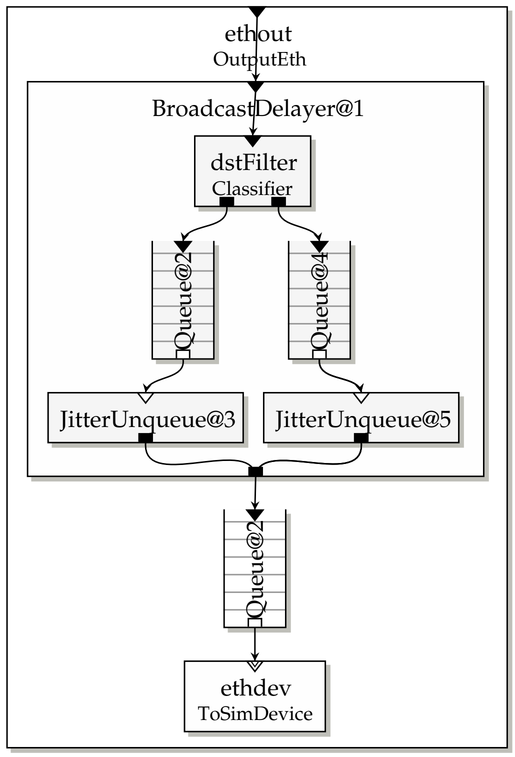
(a) *From host*. `FromSimDevice` pushes out IP packets that the local host wants to transmit using Xcastor. The IP header is marked in the packet and the source address is translated from 127.0.0.1 to the node's external IP address.



(b) *To host*. This `Element` simply delivers packets to the local host.

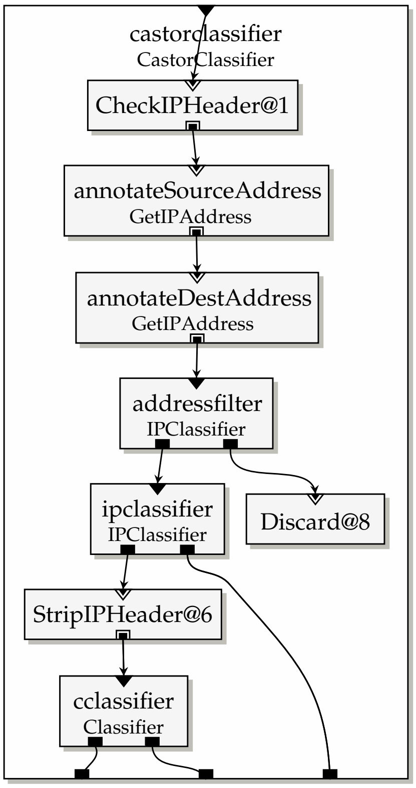


(c) *Ethernet input*. Incoming Address Resolution Protocol (ARP) requests and replies are classified and appropriately processed. Other frames (containing Xcastor PKTs) are directly pushed to the output. Since the `ethdev` is set to promiscuous mode (see Section 5.2.3), no Ethernet filter is applied to Xcastor frames.

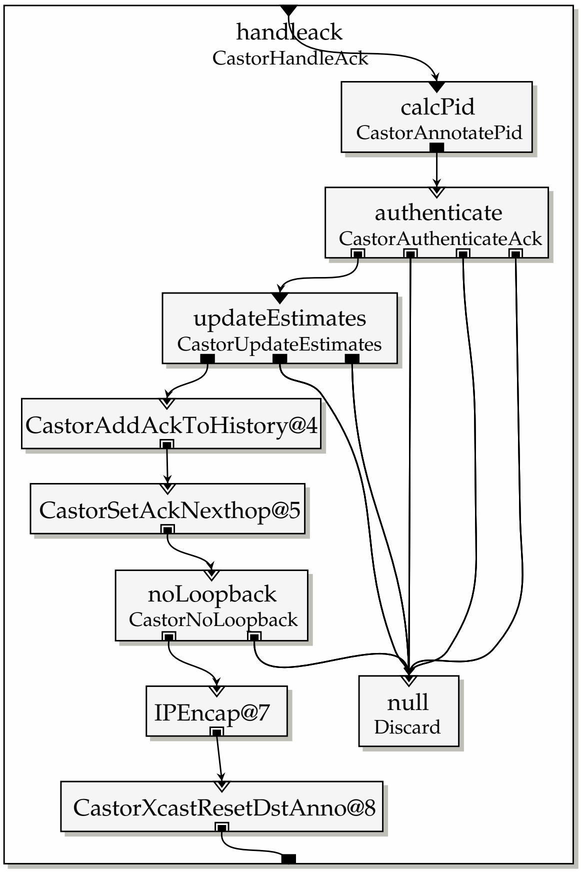


(d) *Ethernet output*. Ethernet frames are pushed to `ethout` and classified based on the destination MAC address. Broadcast frames are delayed as will be discussed in Section 5.2.3.

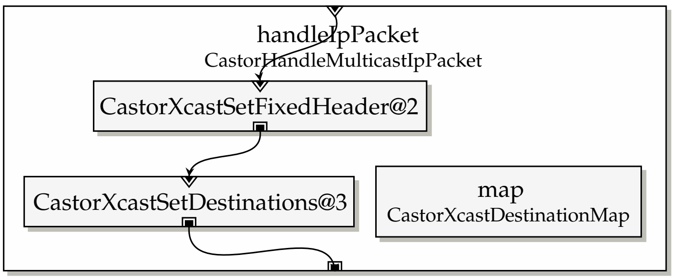
Figure 5.2: Click implementation details: Input/output elements.



(a) *Packet classifier*. Incoming IP packets have their source and destination addresses annotated. The `addressfilter` is required since no Ethernet filter was previously applied (devices operate in promiscuous mode). `CastorClassifier` outputs PKTs, ACKs, and non-Xcastor IP packets.

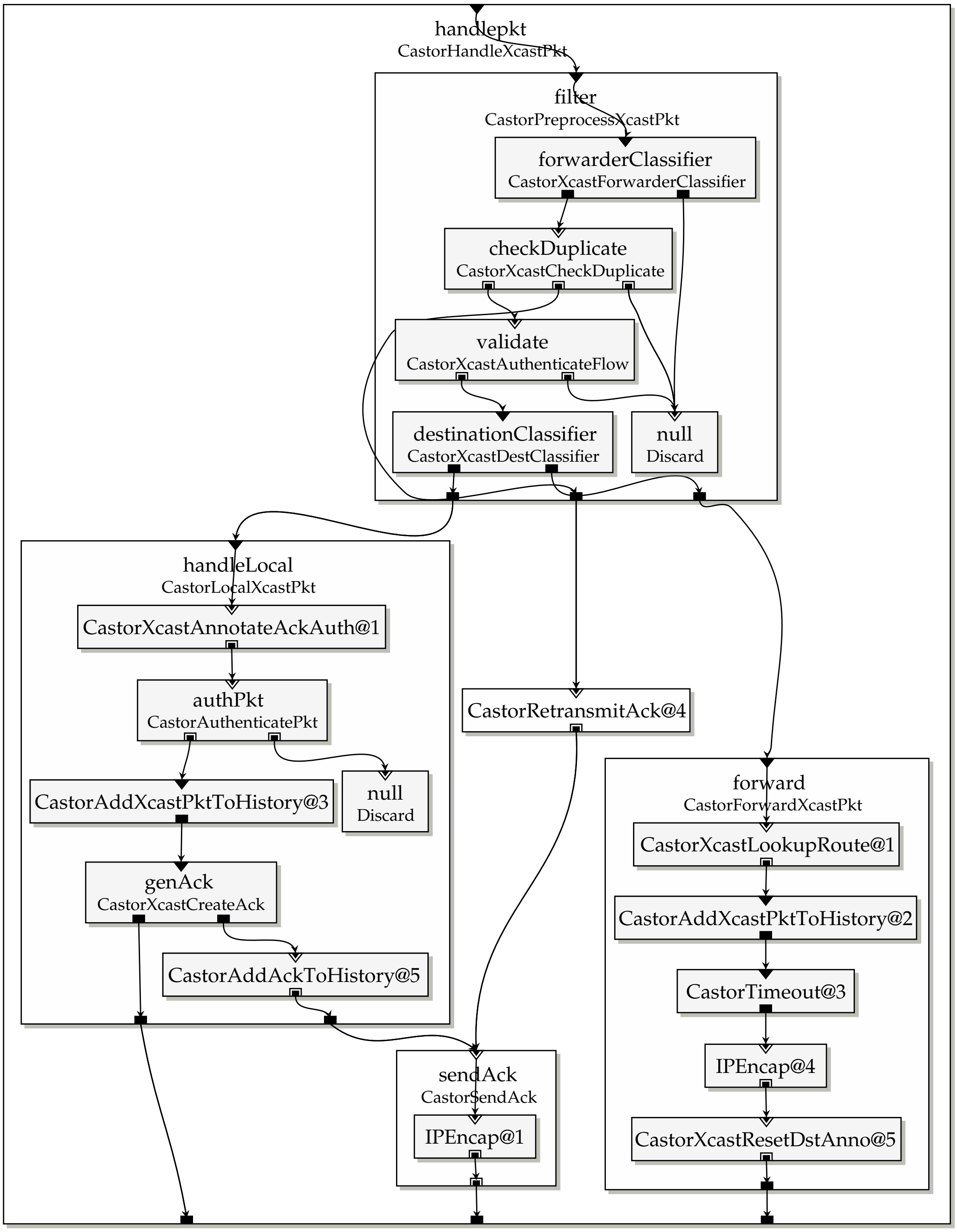


(b) *ACK processing*. The ACK authenticator is hashed to calculate the corresponding PKT identifier. It is then authenticated and eventually used to update the reliability estimators of the appropriate subflow and neighbor. If any test fails, the ACK is discarded. `noLoopback` is used to prevent source nodes from rebroadcasting ACKs that are destined to them. In this implementation, Xcastor packets are wrapped in IP packets to support interoperability with ARP.



(c) *Handle IP packet from host*. The `Element` first prepends the fixed header part (including the Merkle tree logic), and then translates the multicast IP address in the IP packet header to an explicit list of destinations. The individual PKT identifiers are created here as well.

Figure 5.3: Click implementation details: Classifier, ACK processing, and IP packet processing.



The `forwarderClassifier` implements the handling of the forwarder sets and removes the ones that the node itself is not responsible for. The duplicate check removes duplicate PKT identifiers and issues ACK retransmissions if the node already received one. If flow authentication succeeds, the PKT is either delivered (→ `handleLocal`), `forward`ed or both. A local PKT causes the node to authenticate the PKT and, if successful, to add it to the history, to generate an ACK, and to push both the corresponding outputs. When `forward`ing, the next hop is chosen for every destination, each PKT identifier is added to the history, a timer is started, and then the PKT is pushed to the third output. PKTs with multiple receivers are addressed to the broadcast IP address 255.255.255.255 (`IPEncap`). `CastorXcastResetDstAnno` is used to enable MAC layer unicasts for these network layer broadcasts (see Section 5.2.3.1).

Figure 5.4: Click implementation details: PKT processing.

5.2.3 *Interworking with the MAC Layer: Broadcast Reliability*

We rely on IEEE 802.11 broadcasts to transmit Xcastor PKTs to all nodes in the forwarder list. This is a problem when comparing the performance of Xcastor with the original Castor protocol. The 802.11 broadcast mechanism does not support acknowledgments, and by implication, no retransmissions for MAC layer group communication [46]. This has the consequence that Castor performs better in terms of reliability since it has the chance to use MAC layer unicast for PKT transmission more often. To improve Xcastor's performance in this respect we use 802.11 unicasts whenever there is only a single node in the forwarder list. Similarly, ACKs are unicast whenever possible, i. e., *1*) a destination is replying with an ACK to the sender of the PKT; *2*) upon ACK forwarding, if we received the corresponding PKT from a single node; *3*) upon ACK retransmission.

5.2.3.1 *Promiscuous Mode*

To improve link reliability even further, we set the nodes' network interfaces into promiscuous mode: Instead of using MAC layer broadcast when multiple nodes are in the forwarder list, we unicast PKTs to the forwarder that is responsible for most destinations ($\hat{h}$)[3]. The other nodes within transmission range will overhear the unicast PKT and inspect the forwarder list to decide whether to further process the PKT. If $\hat{h}$ was unable to receive the PKT, a retransmission will be issued on the MAC layer, giving all other forwarders a second chance to correctly receive it. These MAC layer unicasts are enabled using a `CastorXcastResetDstAnno Element` (Figure 5.4).

We provide a performance comparison of Xcastor with promiscuous mode enabled and disabled in the next chapter.

5.2.3.2 *Adding Jitter to Broadcast Traffic*

Initial route discovery is negatively affected by frequent broadcasts in the beginning of a communication session, since they do not benefit from MAC layer retransmissions after collisions. To attenuate the effect and avoid concurrent broadcast transmissions, we introduce a `JitterUnqueue Element` in our Click router configuration (Figure 5.2d) that allows us to delay the dissemination of broadcast traffic. The value for the delay is uniformly chosen at random from the interval $[0, \text{Jitter}_{max}]$. Based on the results of Friedman et al. [29], reasonable values for Jitter_{max} are in the order of 100 µs—we set $\text{Jitter}_{max} = 100\,\mu s$ in our experiments.

3 We choose uniformly at random if there exist several $\hat{h}$.

6 EVALUATION

We evaluate our implementation with respect to scalability and security. First, we state the goal of our experiments. We then define our metrics of interest and describe the baseline simulation setup. Finally, we present the simulation results.

6.1 GOALS

We state the intent of our experiments: We investigate the scaling capabilities of Xcastor in Sections 6.4.1 to 6.4.3 by simulating different network sizes, group sizes, and number of groups. Section 6.4.4 addresses the impact of node mobility compared to a static scenario. Eventually, we evaluate the security (attack resilience) of our protocol by placing several blackholes in the network (Section 6.4.5). We summarize the experimental findings in Table 6.1.

EXPERIMENT	SECTION	SUMMARY OF RESULTS
Scalability with respect to network size (number of nodes).	6.4.1	Xcastor operates more efficiently and faster than Castor and flooding in all tested network sizes.
Scalability with respect to group size.	6.4.2	Flooding operates very inefficiently at small group sizes but outperforms Xcastor in terms of bandwidth utilization (BU) at a group size of 10; Xcastor's delay is still significantly lower.
Scalability with respect to number of groups (increased network load).	6.4.3	Castor collapses under higher network load while Xcastor is only marginally affected.
Impact of *mobility*.	6.4.4	Under no mobility, all protocols operate very reliably; flooding performance is not affected by mobility while Castor-like protocols perform ~ 20 % less reliably.
Security: Attack resilience based on the example of blackholes.	6.4.5	Xcastor's reliability is reduced by less than 5 % even in largely hostile environments; Castor suffers from significantly higher packet loss and increased bandwidth utilization at the same setting.

Table 6.1: Evaluation summary: experiments and results.

6.2 METRICS

With the metrics described below, we intend to quantify the reliability (packet delivery rate), efficiency (bandwidth utilization) and speed (delay) of our protocol.

6.2.1 *Packet Delivery Rate*

The packet delivery rate (PDR) indicates how well the routing protocol performs w. r. t. its primary task: reliably delivering Data Packets (PKTs). We count the transmitted messages at the sources and the successfully received ones at the destinations using the individual PKT identifiers (PIDs). For example, a PKT addressed to 5 destinations will be counted as +5 (number of individual PIDs in the PKT) at the source.

$$\text{PDR} = \frac{\text{no. of PIDs received}}{\text{no. of PIDs sent}} \tag{6.1}$$

6.2.2 *Bandwidth Utilization*

Since we are mainly concerned with scalability, the BU, i. e., the total amount of bytes transmitted for delivering a PKT to a single destination is our major interest. The global bandwidth utilization BU_{global} includes all transmissions above the physical layer (Tx). It accounts for PKT and ACK transmissions, Ethernet headers and retransmissions on the MAC layer.

$$BU_{global} = \sum_{i=1}^{n} Tx_i \qquad \text{with } n \text{ nodes in the network.} \tag{6.2}$$

The bandwidth utilization per PID is then calculated as:

$$BU = \frac{BU_{global}}{\text{no. of PIDs sent}} \tag{6.3}$$

6.2.3 *Delay*

The delay is related to the performance of the protocol, i. e., how fast recipients receive their PKTs. We calculate the end-to-end delay of a single PID k as

$$\text{Delay}_k = t_{recv}(k) - t_{send}(k) \tag{6.4}$$

with t_{send} and t_{recv} being the transmission and reception timestamps, respectively. The average delay over all successfully received PIDs[1] is

$$\text{Delay}_{avg} = \frac{1}{N} \sum_{k=1}^{N} \text{Delay}_k. \tag{6.5}$$

1 Failed transmissions have an infinite delay and are thus excluded from the calculation.

6.3 NS-3 DISCRETE EVENT NETWORK SIMULATOR

We select the renowned discrete event network simulator *ns-3* for our evaluation. With the extensions *nsclick* [63] and *ns-3-click* [90], it is possible to run Click configurations in a simulated environment, making use of ns-3's emulated IEEE 802.11 MAC layer [63] and mobility models[2].

6.4 SIMULATION

In Table 6.2, we outline our simulation setup which is based on the setup chosen within the Castor paper [31]. The simulation time is 10 min. Results are averaged over 20 runs. The error bars indicate 95 % confidence intervals. Each run is independently seeded in ns-3 as suggested by [62], with `RngSeed` set to default `12345` and `RngRun` to the run instance $1, \ldots, 20$.

With Table 6.2 as a baseline, we investigate the individual effects of *1*) network size, *2*) group size, *3*) number of groups, *4*) node mobility, and *5*) presence of malicious nodes.

For comparison purposes, we present simulation results showing *1*) Xcastor with and *2*) without promiscuous mode-enabled interfaces, *3*) a Castor implementation that sends multicast packets as multiple unicast packets, and *4*) a flooding protocol that provides no means of security and simply rebroadcasts each packet. The flooding protocol only features a simple duplicate filter. To comply with the notion of PIDs for the metrics' definitions (Section 6.2), a source inserts an (unprotected) unique identifier in every data packet flooded.

6.4.1 *Impact of Network Size*

Network size is a scalability-limiting factor, as we have discussed in Section 2.3. We would like to quantify the impact of different network sizes. We compare the scenarios listed in Table 6.3. The parameters are chosen in such a way that a constant node density ($n/(w \times h)$) and constant average neighbor count[3] of $n \frac{\pi r^2}{w \times h} - 1 \approx 7{,}7$ (ignoring the neighbor count edge effect[4] [62]) are maintained.

The network size has an impact on the PDR of Castor-like protocols: The difference between *small* and *medium* setting is about 5 %. Flooding remains unaffected.

BU grows linearly with the number of nodes when flooding is used (300 % increase from 50 to 200 nodes), while the Castor-like protocols scale sublinearly from 50 to 100 nodes: BU for Xcastor increases by approximately 80 % (both variants), and by 60 % for Castor. At 200 nodes, Castor collapses: Bandwidth utilization is greatly increased and the delay skyrockets beyond 1100 ms. Xcastor scales better, still BU is more than doubled from 100 to 200 nodes.

The delay is lowest for the unoptimized Xcastor both in absolute terms. Flooding is slowest of all protocols by a large (except for Castor at the largest setting): It is more

2 Build instructions for both ns-3 and Click are included in Appendix A.

3 Kurkowski et al. [62] suggest a calculation of the average neighbor count according to $n \frac{\pi r^2}{w \times h}$. We argue that a node is not a neighbor to itself, so we subtract 1.

4 The average neighbor count is reduced for nodes close to the network borders, e. g., a node in a corner of the network area has neighbors only in 25 % of its coverage area.

Network	Dimensions $w \times h$	$3000\,\text{m} \times 3000\,\text{m}$
	Number of nodes n	100
	Transmission Range r	$500\,\text{m}$
Traffic	Sources	4 % (= 4 nodes with 100 nodes in the network)
	Group size	5
	Payload size and rate	256 bytes per 0.25 s
	Castor flow size	256 (flow restart every 64 s)
	Jitter_{max}	$100\,\mu\text{s}$
	MAC layer	IEEE 802.11b at 11 Mbps (unicast and broadcast)
Mobility	Model	Random Waypoint
	Parameters	Velocity: $]0, 20]\,\text{m/s}$, Pause time: 0 s

Table 6.2: Simulation setup: *baseline* configuration.

Number of nodes n	50	100	200
Dimensions $w \times h$ [m^2]	2121×2121	3000×3000	4242×4242
Transmission range r [m]	500	500	500

Table 6.3: Simulation setup: network configurations with a constant node density.

Group size	1	2	5	10
Number of sources	20	10	4	2
Resulting number of Castor flows	20	20	20	20

Table 6.4: Simulation setup: group size configurations with constant number of Castor flows; group size and number of sources are inversely proportional.

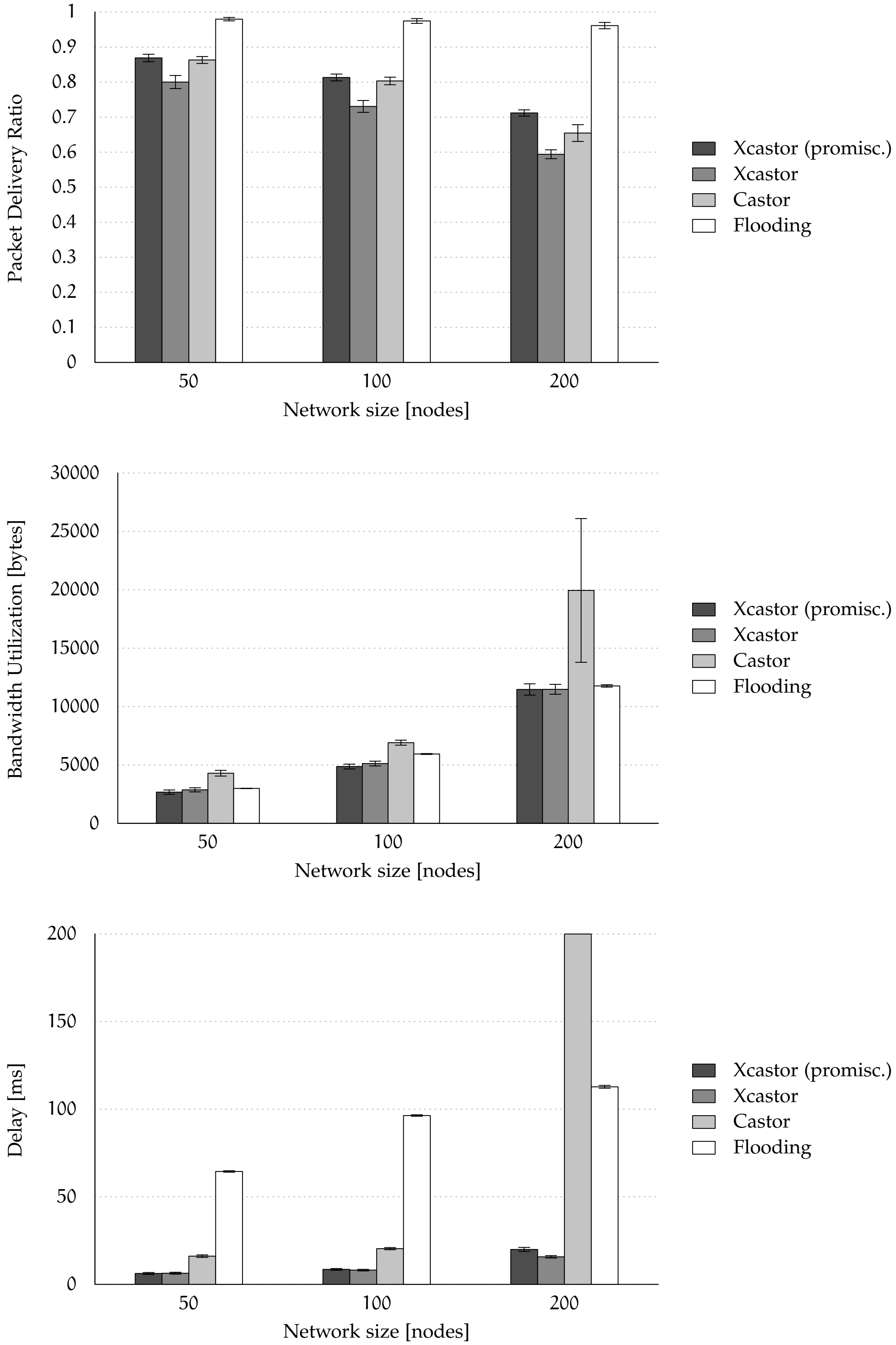


Figure 6.1: Protocol performance with different network sizes. Delay for Castor with 200 nodes skyrockets beyond 1100 ms (clipped at 200 ms for readability).

than 11 times slower than the unoptimized Xcastor at 100 nodes. Castor is slower than Xcastor by a factor of 2.5 (at 50 and 100 nodes).

6.4.2 *Impact of Group Size*

In Section 2.5, we argued that stateless multicast such as Xcast is not suitable for supporting large groups. We want to quantify the limits of our scheme by comparing it to a simple flooding protocol with a negligible header size and a hop count per packet that is close to n.

When changing the group size from the *baseline* setting, we have to be careful not to affect other parameters, e. g., the network load: We adjust the number of groups (i. e., the number of sources) to the group size in such a way that the total source-generated traffic in Castor (i. e., the number of unicast flows) remains constant (see Table 6.4).

As shown in Figure 6.2, all Castor-like protocols perform the same for the unicast setting (group size of 1). PDR remains constant over all group sizes for all protocols except for Xcastor without promiscuous mode enabled (−23 %): MAC layer broadcasts are more frequent in scenarios with larger groups.

Xcastor's BU decreases as group sizes increase, by about 23 % with a group size growing from 1 to 5. The impact of the group size on the flooding protocol is more dramatic: Starting at ~30 000 bytes for unicast, the overhead falls off inversely proportionally to the group size. At a group size of 5 nodes, BU is already close to what is achieved with Xcastor. At 10 nodes, the flooding scheme outperforms all other protocols. The delivery delay for flooding also decreases, but not as quickly as BU: at a group size of 10, flooding is still slower than Xcastor by a factor of 5.2. Interestingly, Castor's performance gets worse with larger groups: Both BU and delay increase, which may be caused by local traffic hot spots around the sources.

6.4.3 *Impact of Number of Groups*

We evaluate the traffic load the protocols are able to handle. Castor collapses under the load of 8 groups, i. e., the network is congested. The result is a vicious circle: Less ACKs are coming through which causes the protocol to broadcast more often, leading to amplified congestion (Figure 6.3). The average delay skyrockets to more than 1200 ms.

The other protocols perform reasonably well under a doubled network load: The average delay for both Xcastor variants is increased by 5 ms and 2.5 ms, respectively; BU increase is 14 % and 6 %, while PDR is reduced by less than 5 %. Flooding remains largely unaffected by the higher load, except for the delay, which is increased by 15 ms.

6.4.4 *Impact of Mobility*

The *baseline* setting with the random waypoint mobility model is compared against a static setting. We show the correctness (PDR close to 1) of all protocols under no mobility. In the mobile case, PDR drops by about 20 % for all Castor-like protocols. Only flooding maintains a PDR close to 1, as to be expected (Figure 6.4).

The results reveal an interesting effect: Both Xcastor variants experience a BU increment of ~45 % while it is less than 10 % for Castor. It shows that Xcastor's advantage

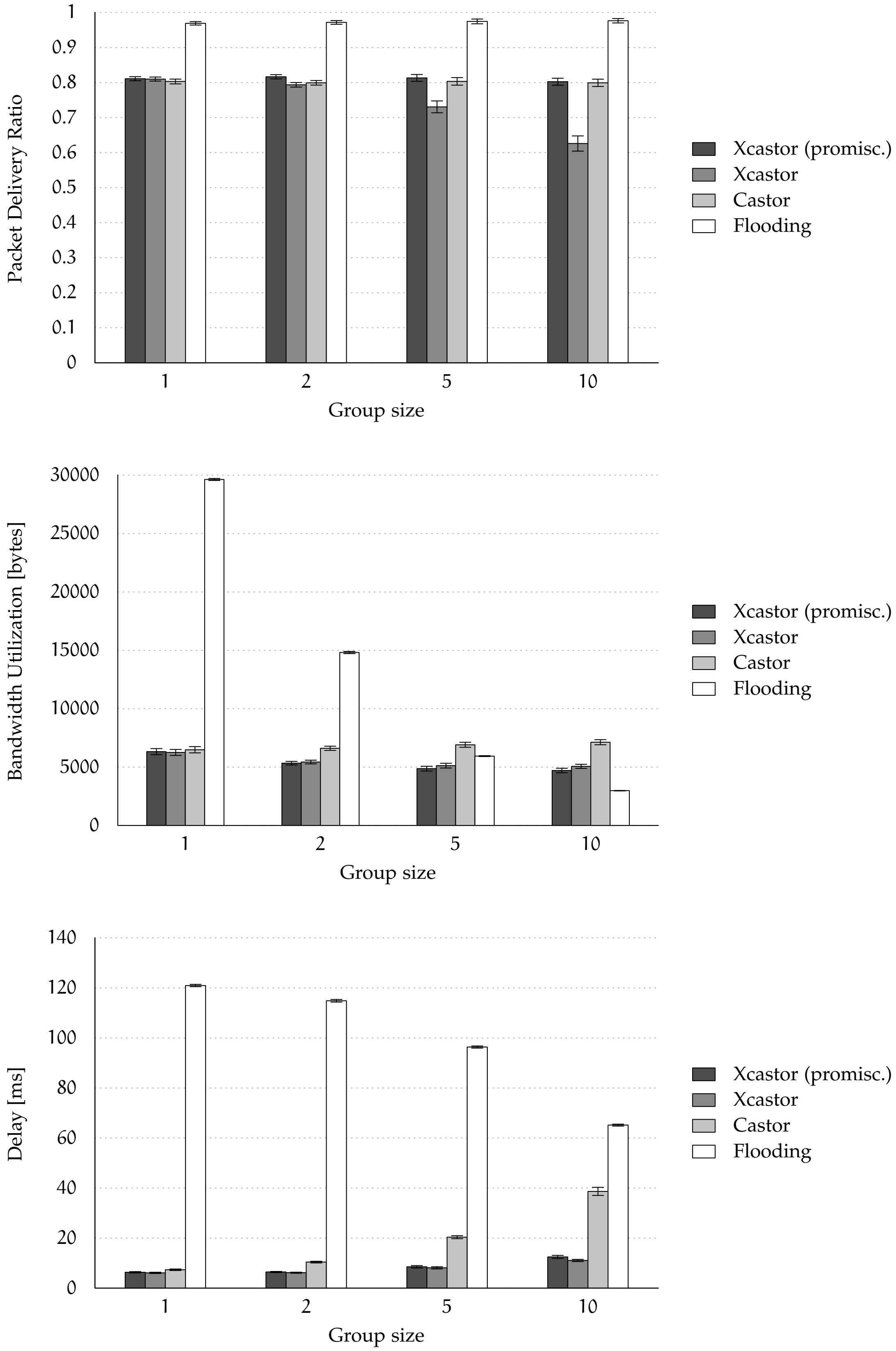


Figure 6.2: Protocol performance with different group sizes.

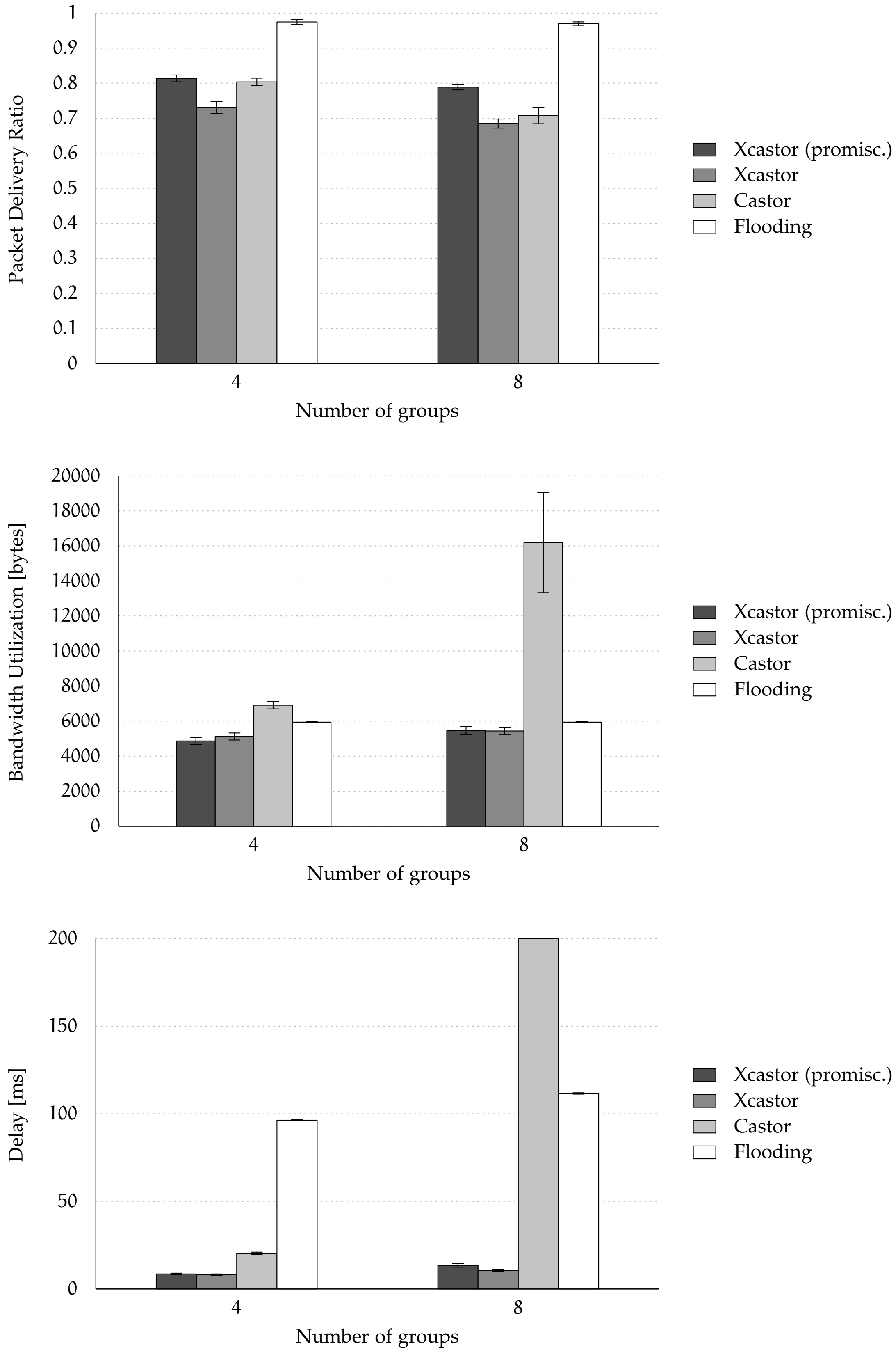


Figure 6.3: Protocol performance with different numbers of groups. Delay for Castor with 8 sources skyrockets beyond 1200 ms (clipped at 200 ms for readability).

over Castor is larger in a static scenario. In the mobile case, Castor-like protocols experience more packet loss, which requires them to broadcast more often. This explains the antithetical behavior of PDR and BU as mobility increases.

Mobility does not have a major impact on the delay of either protocol: There is a slight degradation for all Castor-like protocols possibly due to the increased network load, whereas for flooding the delay remains unaffected.

6.4.5 *Impact of Blackhole Attacks*

So far, we have considered the benign case, i. e., without the presence of an adversary in the network. The attack resistance is evaluated by running the *baseline* setting with different percentages of malicious nodes in the network (Figure 6.5). Note that we consider sources and destinations as always benign in these scenarios; nodes to act as blackholes are chosen uniformly at random from the remaining ones. Malicious nodes conduct a blackhole attack as described in [31], i. e., broadcast PKTs are forwarded to attract traffic while unicast PKTs are dropped. For the flooding protocol, a malicious node simply drops all traffic.

The impact of attackers on both Xcastor variants is relatively small: With 40 % of all nodes as attackers, PDR is reduced by less than 5 %. The success rate for flooding drops by approximately 10 %: Castor suffers the most with a drop of 30 %, at the additional price of an increased bandwidth utilization (+20.7 %).

Xcastor consumes less bandwidth in the adversarial settings. This is due to less nodes forwarding PKTs (blackholes), reducing overall traffic. This effect is more pronounced for flooding: Bandwidth consumption drops by 46 % so that it becomes the most efficient in absolute terms at 40 % blackholes among all protocols. The delay remains unaffected for all protocols.

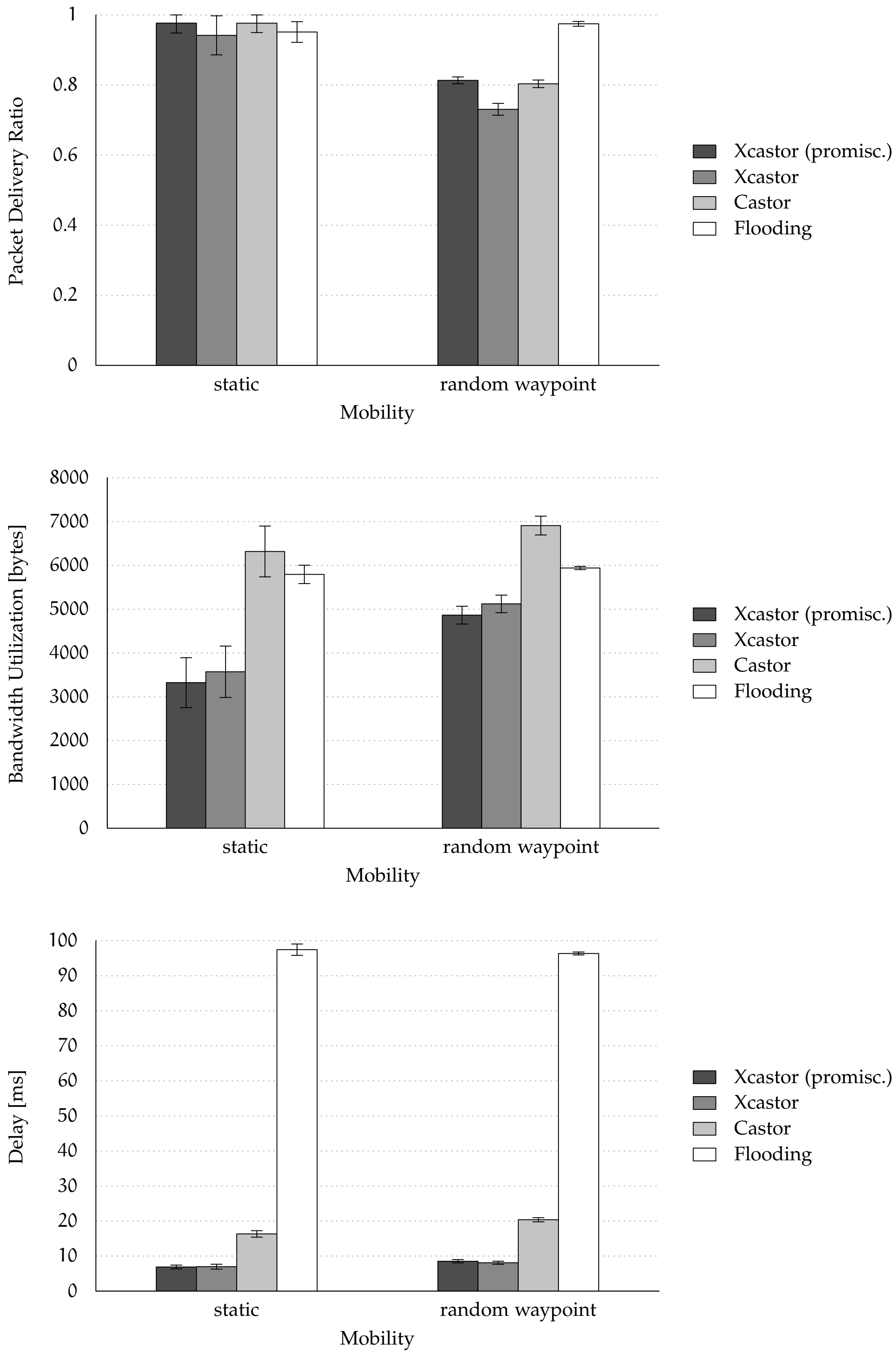


Figure 6.4: Protocol performance without and with mobility.

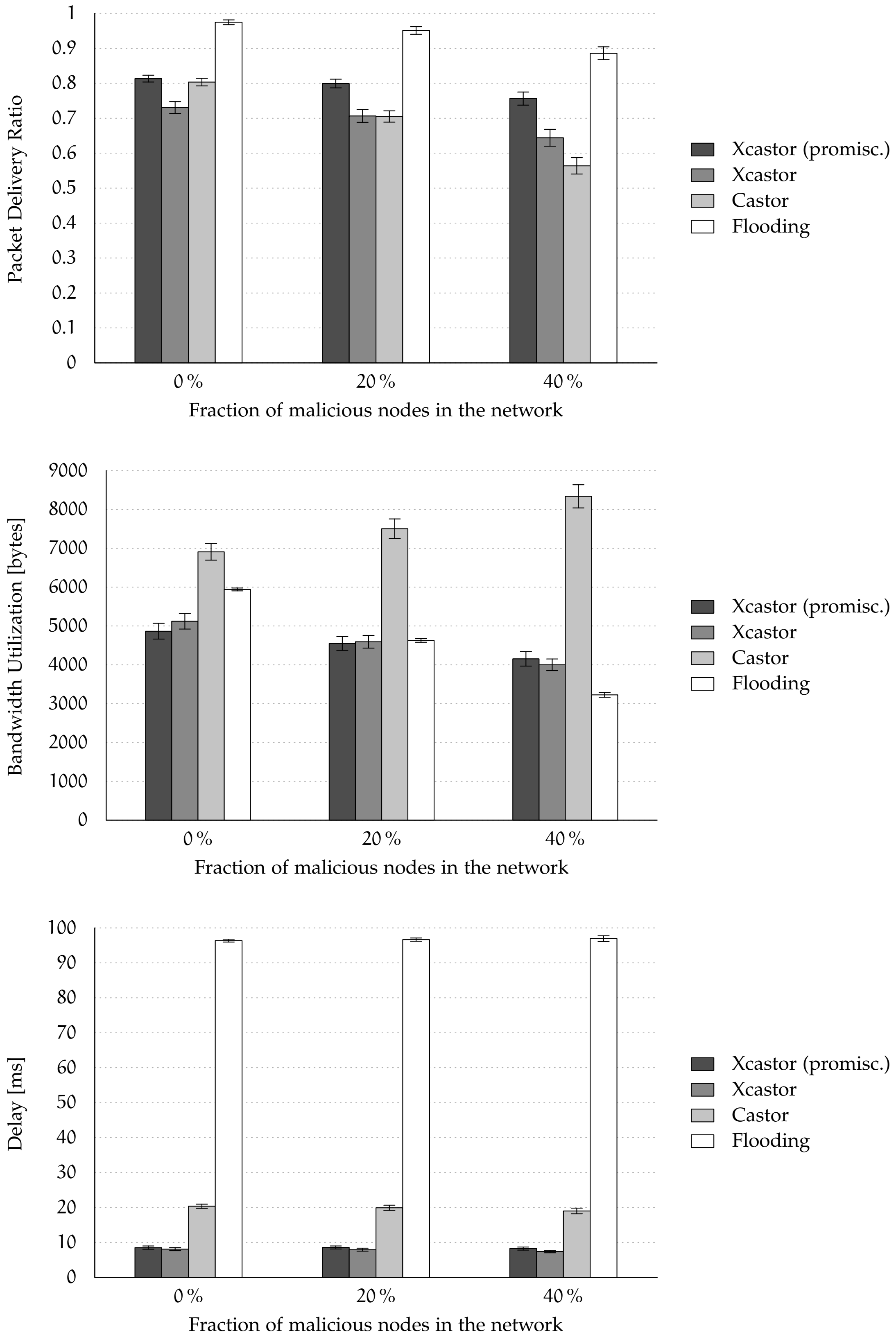


Figure 6.5: Protocol performance under blackhole attacks.

7 DISCUSSION

We discuss the simulation results from Chapter 6: We highlight the strengths of Xcastor compared to the original Castor and a flooding protocol and suggest solutions to further improve its performance.

7.1 XCASTOR HAS LOWEST DELAY

Xcastor is the fastest (lowest delay) of all evaluated protocols. This result is valid for both variants—in all tested settings. Castor is slower than Xcastor except for the unicast setting, where the average delay is almost equal. For Castor, the delay increases considerably with the group size: As the same PKT needs to be transmitted multiple times, traffic "hot spots" are created around the sources (Figure 7.1a), which cause the medium to be busier and consequently result in large MAC-layer backoffs.

Flooding lags far behind the others in terms of delay. There are three reasons for this: *1*) Part of the delay is due to the jitter that we have introduced for broadcast transmissions (Section 5.2.3). In comparison, all Castor-like protocols rely on MAC-layer unicast for most transmissions, so jitter is applied less often. *2*) Network-wide transmissions clog the medium, causing excessive backoffs on the MAC layer. *3*) Packets do not necessarily take the shortest path to the destination: Packets might get lost over the shortest path but arrive at the destination over a different (and potentially longer) one.

7.2 LIMITED BANDWIDTH UTILIZATION GAIN

Applying Xcastor to multicast scenarios clearly decreases the bandwidth utilization needed to serve all of the destinations compared to sending the same packet multiple times to each node using Castor (multicast via unicast).

In the best case, BU gain could grow linearly with the group size (consider a chain-like network scenario where all destinations are connected to the last node in the chain). With our (randomized) network configurations, such linear scaling was not achieved. In our scenarios without any mobility, BU in Xcastor was up to 45 %lower than in Castor. This discrepancy became smaller in mobile scenarios. At the largest tested group size of 10 nodes, the flooding protocol even outperformed Xcastor by 37 %. There are two reasons for the latter observation:

1. The packets sent by Xcastor have approximately triple[1] the size of the actual payload: The major contributors to the large header are the Merkle hash tree and the additional PIDs for every destination.

1 We calculate the upper bound as the maximum possible header size for 10 destinations: $220 + 24 \times 10 + 5 \times 10 = 510$ bytes. With a payload size of 256 bytes, this yields a header to payload ratio of $(510 + 256)/256 \approx 3$.

2. Xcastor PKTs with 10 destinations are forwarded by 50 % of all nodes in our *baseline* scenario: The PKTs are dispersed across the network to reach all randomly placed destinations, which requires a large branching factor of the delivery tree.

Even though flooding outperforms Xcastor in large-group scenarios, it should be noted that the flooding protocol we have evaluated lacks security features such as sender and receiver authentication and acknowledgments. However, the results uncover the weaker spots of Xcastor (header size and branching), which we address in the following.

7.2.1 *Flow Size*

When a sender runs out of PIDs for the current flow—and wishes to continue to transmit PKTs to the same destination(s)—it needs to create a new flow with a new set of PIDs. The new flow has no association with the old flow, which essentially resets the current reliability estimators of forwarding nodes. Subsequent PKTs have to be flooded through the network until (again) a path has been found. The branching factor of the flooded packets is quite high.

It seems reasonable to apply a mechanism that reuses the old routing state in order to avoid information loss. As one possible solution, we suggest to include a "next flow identifier" field in the PKT header such that nodes can initialize the new reliability estimators with the old values.

Interesting questions to answer would be: What is the gain when using such a scheme for long-lived connections? If the flow size is set too small, nodes might miss the next flow identifier field and will have to flood again. If set too large, header size might unnecessarily increase. Finding an optimal flow size could improve the efficiency of both Xcastor and Castor alike.

7.2.2 *Hash Length*

The major contributor to the header size of both Castor and Xcastor is the Merkle hash tree. Reducing the size of the hash values would consequently reduce the header overhead. One option could be to salt the hash calculation and only include and operate on the first half of the hash values. The rationale is that full-size hash values are only required for collision resistance, but the salt allows us to largely ignore collision resistance for two reasons: *1*) only the limited set of hashes within a flow need to be collision-free for the short lifetime of a flow, which was 64 s in our test setting; and *2*) adversaries cannot choose the randomly selected input values of the hash tree, and so they can only try to find another ore-image for a hash value.

7.2.3 *Branching Factor of the Delivery Tree*

Next-hop decisions for a specific destination are agnostic of decisions for other destinations: Xcastor always chooses the most reliable neighbor for each destination. In the worst case, the sender could select n different neighbors as forwarders for n destinations. This leads to essentially n independent flows, which eliminates the advantage over unicast Castor. We demonstrate the effect in a sample protocol run in Figure 7.1b.

We can see that a major portion of the destinations reside in the lower part of the network. The separate route of flow in the upper half of the network is thus unnecessary: The destination at position $(67, 1972)$ could also be served by its neighbor at $(75, 1670)$, which would require only a single additional hop.

Informally, we would like to optimize the next-hop assignment from Equation 4.9 (page 52) of

$$h_i \to \mathcal{F}_i,\ i \in \{1, \ldots, |\mathcal{N}|\}, \qquad \text{with } \mathcal{N} \text{ as the neighbor set}$$

for each PKT in such a way that the number of forwarding nodes

$$\{h_i,\ i \in \{1, \ldots, |\mathcal{N}|\} \ \mid\ \mathcal{F}_i \neq \emptyset\},$$

i. e., the branching factor, is minimal while the overall reliability is maximized.

One such solution could be the inclusion of a third reliability estimator $s^{o}_{H_i,j}$, which is h_j's average reliability to all other destinations. More formally: Let $\mathrm{pkt}(\mathcal{D})$ be the PKT to be forwarded, with $\mathcal{D} = \langle d_1, \ldots, d_i, \ldots, d_n \rangle$ containing the destinations, then calculate

$$s^{o}_{H_i,j} = \frac{1}{|\mathcal{D}|} \sum_{\substack{k=1 \\ k \neq j}}^{|\mathcal{D}|} s_{H_i,k} \qquad \text{and}$$

$$s'_{H_i,j} = \frac{s^{a}_{H_i,j} + s^{f}_{H_i,j} + \eta\, s^{o}_{H_i,j}}{2 + \eta}$$

with η being the parameter controlling the aggressiveness of the optimization (setting $\eta = 0$ ignores the optimization). Finally, select the next hop for d_i as before according to $p_{\max} = \max_j s'_{H_i,j}$.

This metric gives preference to nodes that are reliable forwarders to multiple destinations. Such a forwarder is consequently more likely to be selected for multiple destinations, reducing the branching factor of the PKT.

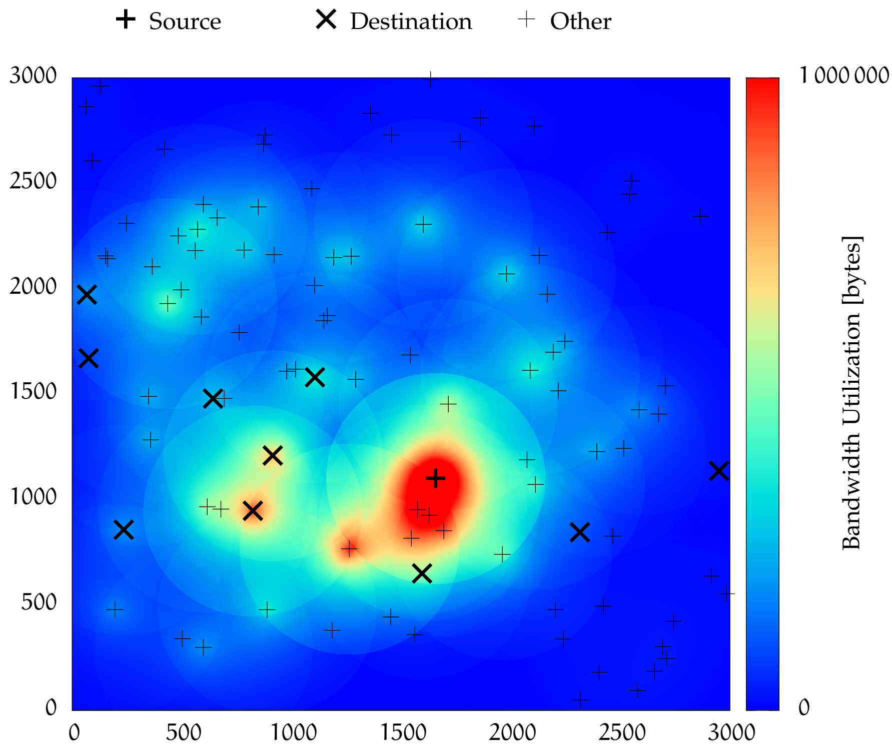
Proper evaluation of this (or another) metric would be time-consuming due to ns-3's simulation times. Thus, we leave it as future work.

7.3 EXPLICIT MULTICAST VS. FLOODING IN LARGE GROUPS

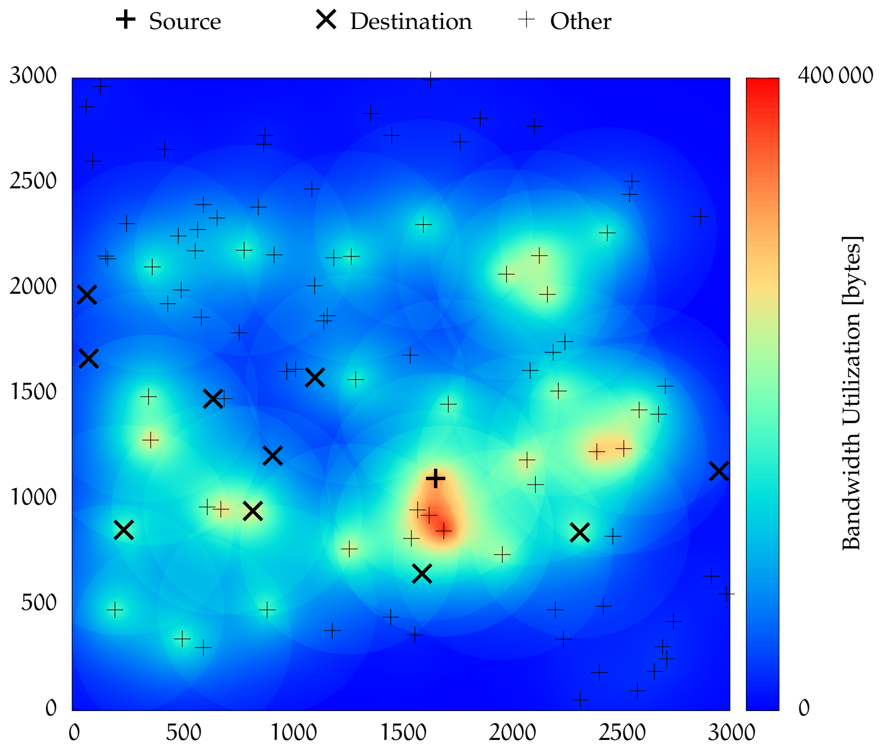
The bandwidth overhead for Xcastor scales linearly with the group size (Figure 6.2b). We want to identify the break-even point where header overhead outweighs the overhead of flooding the entire network. In other words: When is flooding a sensible alternative to Xcast?

Based on our results, this threshold is somewhere between a group size of 5 and 10. We try to gain a deeper understanding on the location of this point.

We construct a simplified analytical equation to describe the break-even point. We assume a perfect MAC layer not requiring retransmissions and ignore Xcastor ACKs for simplicity. We denote $m(\mathcal{D})$ as the number of rebroadcasts for Xcast packets and n



(a) Castor: A traffic hot spot is clearly visible around the sender.



(b) Xcastor: Traffic is more evenly spread across the forwarders.

Figure 7.1: Spatial distribution of bandwidth utilization in Castor and Xcastor. The figures were taken from protocol run # 10 with 100 nodes, 1 sender and a group size of 10 nodes with no mobility and 64 s simulation time (= lifetime of a single flow), but otherwise complying to our *baseline* setting).

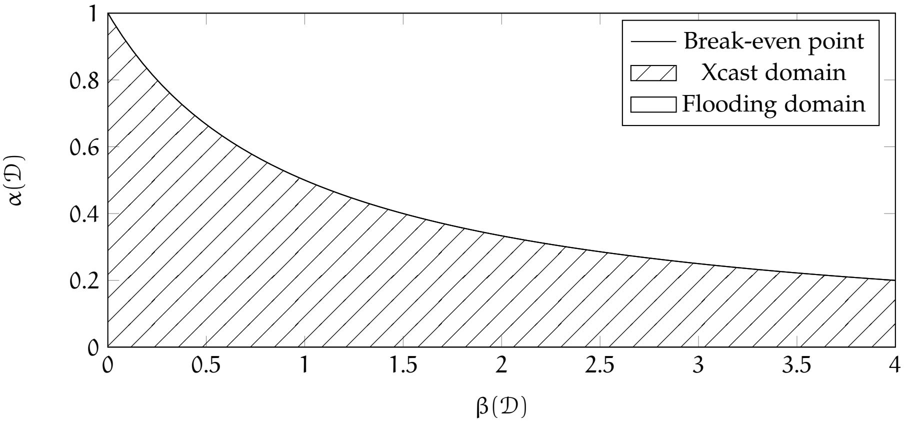


Figure 7.2: Operational domains: Xcast vs. flooding (plot of Equation 7.1)

as the number of rebroadcasts for the flooding protocol (= network size). $\mathcal{H}(\mathcal{D})$ is the group size-dependent Xcast header size and $\mathcal{P}$ the common payload size.

$$\begin{aligned} m(\mathcal{D}) \times (\mathcal{H}(\mathcal{D}) + \mathcal{P}) = n \times \mathcal{P} \qquad & \Longleftrightarrow \\ \frac{m(\mathcal{D})}{n} \times \frac{\mathcal{H}(\mathcal{D}) + \mathcal{P}}{\mathcal{P}} = 1 \qquad & \Longleftrightarrow \\ \underbrace{\frac{m(\mathcal{D})}{n}}_{\alpha(\mathcal{D})} \times \left(\underbrace{\frac{\mathcal{H}(\mathcal{D})}{\mathcal{P}}}_{\beta(\mathcal{D})} + 1 \right) = 1 \qquad & \Longleftrightarrow \\ \alpha(\mathcal{D})\, \beta(\mathcal{D}) + \alpha(\mathcal{D}) - 1 = 0 & \end{aligned} \tag{7.1}$$

Even though Equation 7.1 is only a rough approximation, we can see that the break-even point is not fixed but depends on

$\alpha(\mathcal{D})$: The ratio of rebroadcasts $m(\mathcal{D})$ to the network size n since a flooded packet is rebroadcast n times (this factor is also related to the branching factor discussed in Section 7.2.3); and

$\beta(\mathcal{D})$: The ratio of the group-size-dependent Xcastor header size $\mathcal{H}(\mathcal{D})$ and the size of the payload $\mathcal{P}$.

Figure 7.2 shows that for small payloads, flooding increasingly becomes the better choice. On the other hand, the better Xcast manages to keep the branching factor—and thus, the number of rebroadcasts low–the more likely Xcast is to be preferred.

We give an example for Equation 7.1: Consider the *baseline* setting (Table 6.2) with a group size $|\mathcal{D}| = 10$ and payload size $|\mathcal{P}| = 256$ bytes. Given an upper bound Xcastor header size[2] of 510 bytes, we approximate $\beta(\mathcal{D}) \approx 2$ (neglecting bandwidth utilization from MAC layer retransmissions and ACKs), and, according to Equation 7.1, $\alpha(\mathcal{D}) \approx 0.33$. Empirically, the average hop count for such an Xcastor PKT is 50, that is, half of the nodes in the network participate in PKT forwarding, so $\alpha_{\text{empiric}}(\mathcal{D}) = 0.50$. Since $\alpha_{\text{empiric}}(\mathcal{D}) > 0.33 = \alpha(\mathcal{D})$, flooding should be more efficient at $|\mathcal{D}| = 10$, which is confirmed by our results (Figure 6.2).

2 We calculate the upper bound as the maximum possible header size for 10 destinations: $220 + 24 \times 10 + 5 \times 10 = 510$ bytes.

7.4 MAC LAYER RELIABILITY IS IMPORTANT

During evaluation, it became evident that Xcastor's packet delivery rate largely depends on successful MAC layer transmissions when addressing larger groups (compare promiscuous vs. non-promiscuous mode variants in Figure 6.2). Temporary link breakage due to collisions, nodes not ready to receive, or other reasons can be mitigated by local retransmissions. According to IEEE 802.11 [46], nodes shall attempt up to 7 retransmission[3] for unicast frames until an acknowledgment is received. Broadcast frames are never retransmitted.

7.4.1 *MAC Layer Multicast with Acknowledgments*

Gossain et al. proposed MAC-layer acknowledgments and retransmissions for multicast frames [34]: Acknowledgments are expected from every receiver, otherwise the frame is retransmitted to the non-responding nodes. The authors propose a strict sequential order of ACK transmissions to avoid collisions from multiple receivers sending ACKs at the same time. Since we already saw a reliability improvement for "pseudo-multicast" (promiscuous-mode Xcastor) over normal broadcast, we suggest to evaluate the effect of full MAC-layer multicast support.

ACK delivery could also benefit from such a scheme (both Xcastor and Castor): When forwarding ACKs, Castor relies on MAC layer broadcasts. However, usually not all but only a few neighbors have previously forwarded the corresponding PKT. Thus, MAC-layer multicast could be used to more reliably transmit ACKs to those (few) neighbors.

7.5 ATTACK RESILIENCE IMPROVED

Resilience to the blackhole attack is not negatively affected but actually improved by our Xcastor extension: At 40 % blackholes, Xcastor's PDR is reduced by just less than 5 %, whereas Castor suffers from a 30 % in its PDR. Such a severe impact was not anticipated, especially since Galuba et al. [31] indicated a smaller drop in their results. We assume that the discrepancy is caused by our different traffic setting: Galuba et al. used node-disjoint sender-receiver pairs while in our scenario a Castor source initiates multiple flows. What follows is an increased BU around the sources which causes additional packet loss.

3 `dot11ShortRetryLimit` for frames smaller than the RTS/CTS threshold defaults to 7, while `dot11LongRetryLimit` for larger frames defaults to 4.

8 CONCLUSION

Decentralized ad-hoc networks are on the verge of becoming communication alternatives to centralized systems such as the Internet or the mobile phone cellular network in specific scenarios, e. g., in emergency communication. Due to the unmanaged nature of such networks, secure routing is important to maintain operability of the system. Unfortunately, while secure unicast routing has been a research issue for more than a decade, secure multicast routing, important for reliably addressing groups of receivers, remains understudied.

In this thesis, we first presented the state of the art in scalable and secure MANET routing. Based on the results, we then developed a secure multicast extension for Castor, a promising secure and scalable unicast routing protocol. We call our extension Xcastor. Using an Explicit Multicast-based approach, we enabled secure routing to multiple destinations while reducing the bandwidth consumption by up to 45 % compared to Castor. Castor's performance in terms of its packet delivery rate (PDR) was maintained and, in some scenarios, even slightly improved due to reduced network load. The delivery delay was significantly decreased by a factor of 2.5. By following Castor's *security by design* approach and because of Xcastor's reduced impact on the network load, we achieved a significantly better blackhole attack resilience than the original protocol. Xcastor's performance is identical to Castor when addressing a single destination, which means that Xcastor could replace Castor as a combined unicast and multicast protocol without any loss in efficiency.

However, we also identified room for improvement: Castor's reliability metric, which is used by Xcastor, is multicast-agnostic. We argue that a multicast-aware metric could decrease bandwidth utilization even further with little reduction in overall reliability. In addition, we found that the MAC-layer acknowledgment mechanism has a severe impact on the PDR when the group size is increased.

Flooding is quite inefficient when it comes to unicast routing as every node in the network will rebroadcast the packet. Similarly, it is inefficient when used to address a small number of receivers with a single packet. Instead, an Xcast-based protocol such as Xcastor can provide a much better performance. However, we have shown that there exists a break-even point where the growing Xcast header size outweighs the additional rebroadcasts necessary for flooding. We found that this break-even point is dependent on the payload size as well as the branching factor of the delivery tree.

8.1 OUTLOOK

Based on our discussion in Chapter 7, we suggest future development of Xcastor to be concerned with *1*) developing and evaluating an adapted metric minimizing the size of the forwarder list while maintaining a high degree of reliability; *2*) solving the problem of flow restarts; and *3*) investigating the impact of a MAC-layer retransmission scheme for multicast frames. In addition, evaluating Xcastor in a real testbed would further support the credibility of the obtained simulation results.

Part III

APPENDIX

A BUILD INSTRUCTIONS

For completeness, we include the build instructions for Click and ns-3 in Listings A.1 and A.2, respectively. Building our Castor and Xcastor Click implementations requires the Botan C++ crypto library[1] in version 1.10.

```
# Change to Click directory
$: cd <CLICK_DIR>

# Configure Click as userlevel module, enable local
# modules (Castor), ns-3 and WiFi support
$: ./configure --enable-userlevel --disable-linuxmodule
       --enable-local --enable-nsclick

# Build (when building after changing Click elements,
# run 'make elemlist && make' instead)
$: make
```

Listing A.1: Building Click with ns-3 support

```
# Change to ns-3 directory
$: cd <NS3_DIR>

# Configure ns-3 with Click
$: ./waf configure --with-nsclick=<CLICK_DIR>

# Build ns-3 and run <NS3_DIR>/scratch/<EXPERIMENT>.cc
$: ./waf build
$: ./waf --run=<EXPERIMENT>
```

Listing A.2: Building ns-3 with Click support

1 Webpage of Botan C++ crypto library: `http://botan.randombit.net`

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

WMN	Wireless Mesh Network
MANET	Mobile Ad-Hoc Network
VANET	Vehicular Ad-Hoc Network
DTN	Delay Tolerant Network
GPS	Global Positioning System
DVR	Distance Vector Routing
LSR	Link State Routing
LSU	Link State Update
SR	Source Routing
DSDV	Destination-Sequenced Distance-Vector
AODV	Ad hoc On-Demand Distance Vector
OLSR	Optimized Link State Routing
DSR	Dynamic Source Routing
Castor	Continuously Adapting Secure Topology-Oblivious Routing
MAODV	Multicast Ad Hoc On-Demand Distance Vector
Xcast	Explicit Multicast
SGM	Small Group Multicast
RREQ	Route Request
RREP	Route Reply
PKT	Data Packet
ACK	Acknowledgment
TTL	Time To Live
PID	PKT identifier

DHT	Distributed Hash Table
DoS	Denial of Service
CA	Certificate Authority
SA	Security Association
MAC	Media Access Control
PDR	packet delivery rate
BU	bandwidth utilization

BIBLIOGRAPHY

[1] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly. Denial of service resilience in ad hoc networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 202–215, New York, NY, USA, 2004. ACM.

[2] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[3] Tony Ballardie. Scalable multicast key distribution. RFC 1949, IETF, May 1996.

[4] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core based trees (CBT). In *Conference Proceedings on Communications Architectures, Protocols and Applications*, SIGCOMM, pages 85–95, New York, NY, USA, 1993. ACM.

[5] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 76–84, New York, NY, USA, 1998. ACM.

[6] Mark Baugher, Ran Canetti, Lakshminath R. Dondeti, and Fredrik Lindholm. Multicast security (MSEC) group key management architecture. RFC 4046, IETF, April 2005.

[7] BBC News. Iraqis use Firechat messaging app to overcome net block, June 2014. URL `http://www.bbc.com/news/technology-27994309`. Accessed: August 28, 2014.

[8] Elizabeth M. Belding-Royer and Charles E. Perkins. Evolution and future directions of the ad hoc on-demand distance-vector routing protocol. *Ad Hoc Networks*, 1(1):125–150, 2003.

[9] Matt Bishop. A security analysis of the NTP protocol version 2. In *Proceedings of the 6th Annual Computer Security Applications Conference*, pages 20–29, December 1990.

[10] Stefan Brands and David Chaum. Distance-bounding protocols. In Tor Helleseth, editor, *Advances in Cryptology—EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 344–359. Springer Berlin Heidelberg, 1994.

[11] Levente Buttyan and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007.

[12] Viveck R. Cadambe and Syed A. Jafar. Interference alignment and degrees of freedom of the k-user interference channel. *Information Theory, IEEE Transactions on*, 54(8):3425–3441, August 2008.

[13] Gianni Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *European Transactions on Telecommunications*, 16(5):443–455, 2005.

[14] Kai Chen and Klara Nahrstedt. Effective location-guided tree construction algorithms for small group multicast in MANET. In *Proceedings of the IEEE Conference on Computer Communications*, volume 3 of *INFOCOM*, pages 1180–1189, 2002.

[15] Ching-Chuan Chiang and Mario Gerla. Routing and multicast in multihop, mobile wireless networks. In *IEEE 6th International Conference on Universal Personal Communications Record*, volume 2, pages 546–551, October 1997.

[16] Ching-Chuan Chiang, Mario Gerla, and Lixia Zhang. Forwarding group multicast protocol (FGMP) for multihop, mobile wireless networks. *Cluster Computing*, 1(2): 187–196, 1998.

[17] Citizen Lab. Iraq information controls update: Analyzing internet filtering and mobile apps, June 2014. URL `https://citizenlab.org/2014/07/iraq-information-controls-update-analyzing-internet-filtering-mobile-apps/#part2`. Accessed: August 28, 2014.

[18] Thomas Heide Clausen and Philippe Jacquet. Optimized link state routing protocol (OLSR). RFC 3626, IETF, October 2003.

[19] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4): 419–434, July 2005.

[20] Gergely Ács, Levente Buttyán, and István Vajda. Provably secure on-demand source routing in mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 5(11):1533–1546, November 2006.

[21] Reza Curtmola and Cristina Nita-Rotaru. BSMR: Byzantine-resilient secure multicast routing in multihop wireless networks. *IEEE Transactions on Mobile Computing*, 8(4):445–459, April 2009.

[22] Editor David R. Oran. OSI IS-IS intra-domain routing protocol. RFC 1142, IETF, February 1990.

[23] Carlos de Morais Cordeiro, Hrishikesh Gossain, and Dharma P. Agrawal. Multicast over wireless mobile ad hoc networks: Present and future directions. *IEEE Network*, 17(1):52–59, January 2003.

[24] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, PODC, pages 1–12, New York, NY, USA, 1987. ACM.

[25] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[26] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, March 1983.

[27] John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer Berlin Heidelberg, 2002.

[28] Förderverein Freie Netzwerke e. V. Freifunk web page. URL `http://freifunk.net`. Accessed: August 28, 2014.

[29] Roy Friedman, David Hay, and Gabriel Kliot. Jittering broadcast transmissions in MANETs: Quantification and implementation strategies. Technical report, Department of Computer Science, The Technion—Israel Institute of Technology, 2009.

[30] Wojciech Galuba, Panagiotis Papadimitratos, Marcin Poturalski, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. More on Castor: the scalable secure routing protocol for ad-hoc networks. Technical Report LSIR-REPORT-2009-002, EPFL, 2009.

[31] Wojciech Galuba, Panagiotis Papadimitratos, Marcin Poturalski, Karl Aberer, Zoran Despotovic, and Wolfgang Kellerer. Castor: Scalable secure routing for ad hoc networks. In *Proceedings of the IEEE Conference on Computer Communications*, INFOCOM, pages 1–9, San Diego, CA, USA, March 2010.

[32] Jose Joaquin Garcia-Luna-Aceves and Ewerton L. Madruga. The core-assisted mesh protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1380–1394, August 1999.

[33] Jose Joaquin Garcia-Luna-Aceves and Dhananjay Sampath. Scalable integrated routing using prefix labels and distributed hash tables for MANETs. In *IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, MASS, pages 188–198, October 2009.

[34] Hrishikesh Gossain, Nagesh Nandiraju, Kumar Anand, and Dharma P. Agrawal. Supporting MAC layer multicast in IEEE 802.11 based MANETs: Issues and solutions. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 172–179, November 2004.

[35] Hrishikesh Gossain, Kumar Anand, Carlos Cordeiro, and Dharma P. Agrawal. A scalable explicit multicast protocol for MANETs. *Communications and Networks*, 7 (3):294–306, September 2005.

[36] Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. In *Proceedings of the IEEE Conference on Computer Communications*, volume 3 of *INFOCOM*, pages 1360–1369, April 2001.

[37] Chao Gui and P. Mohapatra. Scalable multicasting in mobile ad hoc networks. In *Proceedings of the IEEE Conference on Computer Communications*, volume 3 of *INFOCOM*, pages 2119–2129, March 2004.

[38] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[39] Zygmunt J. Haas and Marc R. Pearlman. The performance of query control schemes for the zone routing protocol. *IEEE/ACM Transactions on Networking*, 9 (4):427–438, August 2001.

[40] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Transactions on Networking*, 14(3):479–491, June 2006.

[41] Xiaoyan Hong, Kaixin Xu, and Mario Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, 16(4):11–21, July 2002.

[42] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad Hoc Networks*, 1(1): 175–192, 2003.

[43] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless networks. In *Proceedings of the IEEE Conference on Computer Communications*, volume 3 of *INFOCOM*, pages 1976–1986, March 2003.

[44] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM Workshop on Wireless Security*, WiSe, pages 30–40, New York, NY, USA, 2003.

[45] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, January 2005.

[46] IEEE Computer Society. Part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE Std 802.11-2012, March 2012.

[47] IEEE Computer Society. Standard for ethernet. IEEE Std 802.3-2012, December 2012.

[48] Tomasz Imielinski and Julio C. Navas. GPS-based addressing and routing. RFC 2009, IETF, November 1996.

[49] ISO/IEC JTC 1 Information technology. Information processing systems—open systems interconnection—basic reference model—part 2: Security architecture. ISO 7498-2, 1989.

[50] Atsushi Iwata, Ching-Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1369–1379, August 1999.

[51] Philippe Jacquet, Paul Mühlethaler, Thomas Heide Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the IEEE Multi Topic Conference*, INMIC, pages 62–68, 2001.

[52] Lusheng Ji and M. Scott Corson. Differential destination multicast—a MANET multicast routing protocol for small groups. In *Proceedings of the IEEE Conference on Computer Communications*, volume 2 of *INFOCOM*, pages 1192–1201, 2001.

[53] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353 of *The Kluwer International Series in Engineering and Computer Science*, pages 153–181. Springer US, 1996.

[54] David B. Johnson, Yih-Chun Hu, and David A. Maltz. The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4. RFC 4728, IETF, February 2007.

[55] Suyang Ju and Joseph B. Evans. Scalable cognitive routing protocol for mobile ad-hoc networks. In *Proceedings of the IEEE Global Telecommunications Conference*, GLOBECOM, pages 1–6, December 2010.

[56] Luo Junhai, Ye Danxia, Xue Liu, and Fan Mingyu. A survey of multicast routing protocols for mobile ad-hoc networks. *IEEE Communications Surveys Tutorials*, 11 (1):78–91, First Quarter 2009.

[57] Brad Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6^{th} Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 243–254, New York, NY, USA, 2000. ACM.

[58] Sachin Katti, Hariharan Rahul, Wenjun Hu Dina Katabi, Muriel Médard, and Jon Crowcroft. XORs in the air: Practical wireless network coding. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM, pages 243–254, New York, NY, USA, 2006. ACM.

[59] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[60] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[61] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, IETF, February 1997.

[62] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET simulation studies: The incredibles. *ACM SIGMOBILE Mobile Computing and Communications Review*, 9(4):50–61, October 2005.

[63] Mathieu Lacage and Thomas R. Henderson. Yet another network simulator. In *Workshop on ns-2: The IP Network Simulator*, WNS2, New York, NY, USA, 2006. ACM.

[64] Sung-Ju Lee and Mario Gerla. Split multipath routing with maximally disjoint paths in ad hoc networks. In *Proceedings of the IEEE International Conference on Communications*, volume 10 of *ICC*, pages 3201–3205, 2001.

[65] Sung-Ju Lee, William Su, Julian Hsu, Mario Gerla, and Rajive Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Proceedings of the IEEE Conference on Computer Communications*, volume 2 of *INFOCOM*, pages 565–574, 2000.

[66] Sung-Ju Lee, William Su, and Mario Gerla. On-demand multicast routing protocol in multihop wireless mobile networks. *Mobile Networks and Applications*, 7(6):441–453, 2002.

[67] Adrian Loch, Thomas Nitsche, Alexander Kuehne, Matthias Hollick, Joerg Widmer, and Anja Klein. Practical interference alignment in the frequency domain for OFDM-based wireless access networks. In *International Symposium on a World of Wireless, Mobile and Multimedia Networks*, WoWMoM, June 2014.

[68] Jun Luo, Patrick Th. Eugster, and Jean-Pierre Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. In *Proceedings of the IEEE Conference on Computer Communications*, volume 3, pages 2229–2239, March 2003.

[69] Mahesh K. Marina and Samir R. Das. On-demand multipath distance vector routing in ad hoc networks. In *Proceedings of the* 9^{th} *IEEE International Conference on Network Protocols*, ICNP, pages 14–23, November 2001.

[70] John M. McQuillan, Ira Richer, and Eric C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, 28(5):711–719, May 1980.

[71] John Moy. OSPF version 2. RFC 2178, IETF, April 1998.

[72] Stephen Mueller, Rose P. Tsang, and Dipak Ghosal. Multipath routing in mobile ad hoc networks: Issues and challenges. In Maria Carla Calzarossa and Erol Gelenbe, editors, *Performance Tools and Applications to Networked Systems*, volume 2965 of *Lecture Notes in Computer Science*, pages 209–234. Springer Berlin Heidelberg, 2004.

[73] Axel Neumann, Corinna Aichele, Marek Lindner, and Simon Wunderlich. Better approach to mobile ad-hoc networking (B.A.T.M.A.N.). Internet-Draft draft-openmesh-b-a-t-m-a-n-00, IETF, March 2008.

[74] Hoang Lan Nguyen and Uyen Trang Nguyen. A study of different types of attacks on multicast in mobile ad hoc networks. *Ad Hoc Networks*, 6(1):32–46, 2008.

[75] Open Garden. FireChat web page. URL `https://opengan.com/firechat`. Accessed: August 28, 2014.

[76] Open-mesh. B.A.T.M.A.N. protocol concept. URL `http://www.open-mesh.org/projects/open-mesh/wiki/BATMANConcept`. Accessed: May 21, 2014.

[77] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Commnication Networks and Distributed Systems Modeling and Simulation Conference*, pages 193–204, San Antonio, TX, USA, January 2002.

[78] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure link state routing for mobile ad hoc networks. In *Symposium on Applications and the Internet Workshops*, pages 379–383, 2003.

[79] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1):193–209, July 2003.

[80] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure data communication in mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2): 343–356, February 2006.

[81] Panagiotis Papadimitratos and Aleksandar Jovanovic. GNSS-based positioning: Attacks and countermeasures. In *Proceedings of the IEEE Military Communications Conference*, MILCOM, pages 1–7, 2008.

[82] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. A wireless hierarchical routing protocol with group mobility. In *IEEE Wireless Communications and Networking Conference*, volume 3 of *WCNC*, pages 1538–1542, 1999.

[83] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing: A routing scheme for ad hoc wireless networks. In *IEEE International Conference on Communications*, volume 1 of *ICC 2000*, pages 70–74, 2000.

[84] Guangyu Pei, Mario Gerla, and Xiaoyan Hong. LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility. In *Proceedings of the 1^{st} ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc, pages 11–18, Piscataway, NJ, USA, 2000.

[85] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, SIGCOMM, pages 234–244, New York, NY, USA, 1994. ACM.

[86] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Second IEEE Workshop on Mobile Computing Systems and Applications*, WMCSA, pages 90–100, February 1999.

[87] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, IETF, July 2003.

[88] Jon Postel. Transmission control protocol. RFC 793, IETF, September 1981.

[89] Marcin Poturalski, Panagiotis Papadimitratos, and Jean-Pierre Hubaux. Towards provable secure neighbor discovery in wireless networks. In *Proceedings of the 6^{th} ACM Workshop on Formal Methods in Security Engineering*, FMSE, pages 31–42, New York, NY, USA, 2008. ACM.

[90] Lalith Suresh Puthalath and Ruben Merz. NS-3-Click: Click modular router integration for NS-3. In *Proceedings of the 4^{th} International ICST Conference on Simulation Tools and Techniques*, SIMUTools, pages 423–430, ICST, Brussels, Belgium, Belgium, 2011. ICST.

[91] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Rapport de recherche RR-3898, INRIA, 2000.

[92] Bo Rong, Hsiao-Hwa Chen, Yi Qian, Kejie Lu, Rose Qingyang Hu, and Sghaier Guizani. A pyramidal security model for large-scale group-oriented computing in mobile ad hoc networks: The key management study. *IEEE Transactions on Vehicular Technology*, 58(1):398–408, January 2009.

[93] Sankardas Roy, Venkata Gopala Krishna Addada, Sanjeev Setia, and Sushil Jajodia. Securing MAODV: Attacks and countermeasures. In *SECON*, pages 521–532, 2005.

[94] Elizabeth M. Royer and Charles E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Proceedings of the 5th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 207–218, New York, NY, USA, 1999. ACM.

[95] Dhananjay Sampath and Jose Joaquin Garcia-Luna-Aceves. PROSE: Scalable routing in MANETs using prefix labels and distributed hashing. In *6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, SECON, pages 1–9, June 2009.

[96] César A. Santiváñez, Ram Ramanathan, and Ioannis Stavrakakis. Making link-state routing scale for ad hoc networks. In *Proceedings of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc, pages 22–32, New York, NY, USA, 2001.

[97] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 78–87, November 2002.

[98] Jaydip Sen. Security and privacy issues in wireless mesh networks: A survey. In Shafiullah Khan and Al-Sakib Khan Pathan, editors, *Wireless Networks and Security*, Signals and Communication Technology, pages 189–272. Springer Berlin Heidelberg, 2013.

[99] Sudipta Sengupta, Shravan Rayanchu, and Suman Banerjee. Network coding-aware routing in wireless networks. *IEEE/ACM Transactions on Networking*, 18 (4):1158–1170, August 2010.

[100] Thrasyvoulos Spyropoulos, Rao Naveed Rais, Thierry Turletti, Katia Obraczka, and Athanasios Vasilakos. Routing for disruption tolerant networks: Taxonomy and design. *Wireless Networks*, 16(8):2349–2370, November 2010.

[101] Athichart Tangpong, George Kesidis, Hung yuan Hsu, and Ali Hurson. Robust sybil detection for MANETs. In *Proceedings of 18th Internatonal Conference on Computer Communications and Networks*, ICCCN, pages 1–6, August 2009.

[102] Tzeta Tsao, Roger K. Alexander, Mischa Dohler, Vanesa Daza, and Angel Lozano. A security threat analysis for routing protocol for low-power and lossy networks (RPL). Internet-Draft draft-ietf-roll-security-threats-10, IETF, September 2014.

[103] Kamin Whitehouse, Alec Woo, Fred Jiang, Joseph Polastre, and David Culler. Exploiting the capture effect for collision detection and recovery. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 45–52, 2005.

[104] Bing Wu, Jie Wu, Eduardo B. Fernandez, Mohammad Ilyas, and Spyros Magliveras. Secure and efficient key management in mobile ad hoc networks. *Journal of Network and Computer Applications*, 30(3):937–954, August 2007.

[105] Jason Xie, Rajesh R. Talpade, Anthony McAuley, and Mingyan Liu. AMRoute: Ad hoc multicast routing protocol. *Mobile Networks and Applications*, 7(6):429–439, December 2002.

[106] Dingwen Yuan, Michael Riecker, and Matthias Hollick. Making 'glossy' networks sparkle: Exploiting concurrent transmissions for energy efficient, reliable, ultra-low latency communication in wireless control networks. In Bhaskar Krishnamachari, Amy L. Murphy, and Niki Trigoni, editors, *Wireless Sensor Networks*, volume 8354 of *Lecture Notes in Computer Science*, pages 133–149. Springer International Publishing, 2014.

[107] Manel Guerrero Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the 1st ACM Workshop on Wireless Security*, WiSe, pages 1–10, New York, NY, USA, 2002.

[108] Nianjun Zhou, Huaming Wu, and Alhussein A. Abouzeid. Reactive routing overhead in networks with unreliable nodes. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, MobiCom, pages 147–160, New York, NY, USA, 2003. ACM.