N-Tube: Formally Verified Secure Bandwidth Reservation in Path-Aware Internet Architectures

Thilo Weghorn* Swisscom thilo.weghorn@swisscom.com
Si Liu* ETH Zurich si.liu@inf.ethz.ch
Christoph Sprenger ETH Zurich sprenger@inf.ethz.ch
Adrian Perrig ETH Zurich adrian.perrig@inf.ethz.ch
David Basin ETH Zurich basin@inf.ethz.ch

***Abstract*—We present N-Tube, a novel, provably secure, inter-domain bandwidth reservation algorithm that runs on a network architecture supporting path-based forwarding. N-Tube reserves global end-to-end bandwidth along network paths in a distributed, *neighbor-based*, and *tube-fair* way. It guarantees that benign bandwidth demands are granted *available* allocations that are *immutable*, *stable*, *lower-bounded*, and *fair*, even during adversarial demand bursts.**

We formalize N-Tube and powerful adversaries as a labeled transition system, and inductively prove its safety and security properties. We also apply statistical model checking to validate our proofs and perform an additional quantitative assessment of N-Tube, providing strong guarantees for protection against DDoS attacks. We are not aware of any other complex networked system designs that have been subjected to a comparable analysis of both their qualitative properties (such as correctness and security) *and* their quantitative properties (such as performance).

I. Introduction

Providing useful guarantees during DDoS attacks remains an open problem. The increasing sophistication of attacks has not yet been countered by progress in scalable, cost-effective defenses. Sophisticated attacks do not target the victim directly, but just a few critical network links carrying the victim's traffic. For example, in Crossfire, a botnet sends low-volume flows to public servers that are chosen to flood critical links required for the victim's traffic [1]. Similarly, in Coremelt, an adversary sets up traffic flows among pairs of bots that it controls in a way that floods critical links [2]. In these strongest known attacks, an attacker with limited resources can effectively attack critical links and degrade connectivity for large Internet regions [3]. Current techniques cannot defend against such attacks since the congestion hotspots are outside the victims' control.

DDoS protection can be realized by an effective quality of service (QoS) scheme that provides hard bandwidth guarantees in the face of sophisticated adversaries. Since best-effort delivery and over-provisioned network bandwidth enable good performance in the average case, offering QoS guarantees requires fair resource allocation even when bandwidth becomes scarce [4]. Previous QoS architectures, such as IntServ [5], DiffServ [6], and RSVP [7] were designed for the Internet with trusted network participants, not for adversarial scenarios. It remains an open research problem *how to allocate bandwidth in malicious contexts such that legitimate hosts obtain useful bandwidth guarantees.*

A core challenge is that current link-flooding attacks can be caused by a huge number of low-volume flows originating from colluding legitimate-looking bots, e.g., as seen in the Hidden Cobra DDoS Botnet Infrastructure [8]. Therefore, standard fairness notions that QoS solutions try to achieve, such as per source [9], per destination [10], per flow [11], per computation [12], and per class [13], are insufficient in such settings and result in unfair bandwidth allocations. These fairness notions suffer from the "tragedy of the commons" [14], whereby the incentive of rational agents to increase their share of a commonly available resource leads to infinitesimally small shares for less aggressive, honest agents. In particular, in today's Internet, congestion-control-based fairness is the most commonly used *per-flow fairness* notion, which allows adversarial agents to request arbitrarily many flows and thereby obtain a disproportional amount of bandwidth compared to honest agents [15]. Moreover, current QoS architectures lack packet authentication, which is required to monitor and enforce the allocated bandwidth in the presence of malicious agents.

Secure Bandwidth Allocation. From the discussion above, we extract the following two main requirements for secure bandwidth allocation. First, we need a suitable notion of fairness for adversarial settings, and second, we seek to provide a minimal bandwidth guarantee to honest agents, even in the presence of excessive adversarial demands. Moreover, given the complexity of bandwidth allocation algorithms and the unpredictability of adversarial behaviors, we provide the formal specification and verification of their desired properties. There is currently no proposal satisfying these requirements.

In contrast to the current Internet, new Internet architectures supporting *path-based forwarding* provide the prerequisites to achieve these requirements. Instead of using frequently updated forwarding tables, as in today's Internet, they leverage path-based forwarding where the paths taken by data packets stay fixed and correspond to the reserved paths. This simplifies reasoning about resource allocation. SCION [16], NEBULA [17], Pathlets [18], and NIRA [19] are prominent examples of such architectures, where the first already sees real-world deployment [20], [21]. Moreover, SCION and ICING [22] (which is part of the NEBULA architecture) already include packet authentication, which is needed for monitoring and

*Joint first authors. The main part of this work was done while Thilo Weghorn was at ETH Zurich.

enforcing the correct use of allocated bandwidth.

N-Tube Algorithm. We present *N-Tube*, a new *Neighbor-based*, *Tube-fair* bandwidth reservation algorithm, designed to achieve the above requirements. N-Tube introduces a novel notion of allocation fairness called *bounded tube fairness*. N-Tube is designed for networks that support path-based forwarding and prevents link congestion attacks, including the strongest known attacks like Coremelt and Crossfire. It is thus also robust against standard link-flooding attacks, including amplification attacks. To allocate bandwidth on a path, each on-path *autonomous system* (AS) computes and allocates bandwidth locally while accounting for other reservations.

N-Tube builds on two key ideas. First, to always enable the allocation of some (non-zero) bandwidth, N-Tube only uses a fixed fraction of the available bandwidth, saving the rest for future reservation requests. By guaranteeing that the reserved bandwidth stays unchanged until expiry, N-Tube also enables a predictable stabilization period for the bandwidth allocations during times of stable bandwidth demands.

Second, with *bounded tube fairness*, each AS's aggregated bandwidth demands are first *bounded* by the available bandwidth, and then split proportionally among its immediate network neighbors. Hence, if a malicious AS (outside the honest path) tries to congest a link, the first honest AS between the attacker and targeted link limits the adversarial demands, thereby preventing it from obtaining a disproportional share of bandwidth on that link. Consequently, N-Tube also guarantees any honest source AS a *lower bound on the allocated bandwidth*, independently of the desired destination.

Verification Approach. Inter-domain bandwidth reservation is, in general, a difficult problem with complex bandwidth allocation dynamics especially for operation in adversarial environments. This necessitates the verification of any proposed bandwidth reservation algorithm to validate its intended properties, in particular to establish both qualitative correctness and security guarantees as well as quantitative guarantees about the system's bandwidth allocation dynamics. The verification of N-Tube's qualitative and quantitative properties is particularly challenging for several reasons: its desired properties must hold in the presence of a powerful adversary and for arbitrary network topologies. Moreover, the model involves unbounded state information and the verification requires non-linear arithmetical reasoning about bandwidth allocation. These features are notoriously hard to handle for automated verification tools.

We tackle this problem by using a combination of mathematical proofs for the qualitative properties and statistical verification and estimation for the quantitative properties. For qualitative guarantees, we verify N-Tube's correctness and security by: (i) formalizing the algorithm, together with the network environment and attackers, as a *labeled transition system* (LTS), (ii) specifying the safety and security properties as predicates over LTS executions, and (iii) proving by *induction* using careful pencil-and-paper proofs that the formal model satisfies these properties.

For quantitative guarantees, we analyze N-Tube's stability and fairness properties using statistical model checking (SMC) [23]. SMC has been successfully used to analyze large-scale distributed systems and has demonstrated its predictive power when used in early design stages, i.e., its estimations are consistent with implementation-based evaluations under realistic deployment [24], [25]. SMC samples and analyzes system executions until a given confidence level is reached. We transform our LTS model into a *probabilistic rewrite theory* for the SMC-based analysis of N-Tube's quantitative properties using the Maude ecosystem [26]. Unlike in implementation-based evaluations, this allows us to explore the large parameter space, to consider various (malicious) scenarios, and to obtain statistics with a desired confidence level and error margin. With our SMC analysis, we also obtain additional confidence in our inductive proofs of N-Tube's qualitative properties.

In networking, formal methods have been applied to verify qualitative and quantitative properties of routing protocols and DoS protection mechanisms. We will discuss this and additional related work in Section VII. However, we are not aware of any prior work that formally models and verifies a bandwidth reservation system, neither in benign nor in adversarial settings. The full formal definitions and proofs, as well as the source code of our development, are available online [27].

Main Contributions. We provide: (i) the first principled solution to the global inter-domain bandwidth allocation problem that offers stable, lower-bounded, and fair bandwidth allocation in adversarial settings (Sections III and IV); (ii) the formalization of N-Tube, a strong attacker model, and all its safety and security properties, as well as inductive proofs establishing these properties (Sections IV-E and V); (iii) the automated statistical verification and estimation of N-Tube's behaviors, both to validate our proofs and to provide quantitative guarantees and assess N-Tube's resistance to attacks in various malicious scenarios (Section VI).

II. Preliminaries

A. Design Goal and Properties

Our goal is to design a provably secure bandwidth reservation architecture that provides hard, worst-case bandwidth guarantees to source ASes for reaching their destination ASes. A key component of such a QoS architecture is a *bandwidth reservation mechanism* that allocates bandwidth according to the demands of source ASes and guarantees a minimum bandwidth allocation even under heavy congestion or flooding attacks. Thus, N-Tube should satisfy the following properties:

- G1 **Availability**: Any successful reservation request can reserve bandwidth, in spite of network congestion.
- G2 **Immutability**: The allocated bandwidth of any existing reservation stays fixed until it expires.
- G3 **Stability**: In periods of steady and constant demand, the bandwidth allocation in the entire network stabilizes in a predictable period of time.
- G4 **Minimum Bandwidth Guarantee**: After the network stabilizes, there is a lower bound on the allocated bandwidth, i.e., a minimal bandwidth guarantee even with high external demands such as link-flooding attacks.

G5 **Bounded Tube Fairness**: Bandwidth allocation is distributed proportionally to the requested demands, however, adjusted to the maximally available bandwidth.

Additional requirements ensure that N-Tube is efficient and practical from an operational perspective, see [27, Appendix A].

B. Model and Assumptions

Network Model. We model the network as a connected graph with weighted, directed edges. Nodes in the graph represent the ASes' network interfaces and directed edges denote physical links between these interfaces. Each link starts at an egress interface of an AS, called an *egress link* of the AS, and ends at an ingress interface of another AS, called an *ingress link* of that AS. Each edge has a weight that corresponds to the link's capacity. Using interfaces instead of multiple edges between ASes provides a simple graph, instead of an equivalent multigraph, which is closer to N-Tube's specification. Intra-AS links are not modeled but are assumed to provide sufficient capacity. Given this network structure, we make some additional assumptions.

N1 We assume an inter-domain *control plane* that implements a path discovery protocol, enabling each AS to obtain multiple loop-free paths to reach a destination AS (see, for instance, [16, Chapter 7]).

These paths are expressed at the granularity of interfaces between the ASes.

N2 We assume there are mechanisms to quickly detect link and node failures, and provide alternative paths.

For instance, by frequently running the path discovery protocol, we can ensure a timely provision of alternative paths. Modern architectures like SCION support simultaneous communication over multiple paths, hence providing inherent fault tolerance. Based on assumption **N2**, we consider failure detection and handling as orthogonal to the bandwidth reservation algorithm itself. For more details, see Appendix A.

N3 We assume that clocks are loosely, globally synchronized, i.e., with a time discrepancy between ASes on the order of 100 ms, in contrast to reservation times on the order of minutes.

Since clock synchronization is several orders of magnitude more precise than reservation times, we will approximate these synchronized clocks by a global clock in our model.

Attacker Model. We call an AS *honest* if it follows the protocol, and *compromised* or *malicious* otherwise. We will describe malicious ASes' capabilities below. For a given legitimate reservation request, we distinguish *off-path* and *on-path* ASes.

A1 Any off-path AS may be compromised. Compromised ASes can collude (e.g., as part of a botnet) and attempt to allocate excessive amounts of bandwidth in order to exhaust the available bandwidth.

There is no constraint on the distribution of compromised ASes in the network. Compromised ASes may attempt to request excessive bandwidth through multiple reservations over one or more paths, thus preempting other ASes from obtaining a fair share of the available bandwidth.

A2 Compromised off-path ASes can (i) observe all reservation requests sent to them, (ii) change any unauthenticated fields in such reservation messages, and (iii) inject such modified messages into neighboring links.

This means that attackers cannot defeat the cryptography used to realize message authentication. Hence, attackers can at best replay legitimate reservation requests, possibly modifying their unprotected fields, but they cannot craft new ones for ASes they do not control, e.g., by spoofing a signed message without an appropriate private key.

A3 All on-path ASes are honest.

Honest ASes are expected to refrain from allocating excessive bandwidth due to the associated costs. We cannot allow compromised on-path ASes, as they could insert bogus information in reservation requests, making it impossible to achieve our desired properties, in particular (G1) and (G4). Moreover, they could execute DoS attacks by ignoring reservation requests.

Our modeling is at the granularity of ASes and excludes end hosts within ASes. In particular, we consider the case of malicious end hosts within off-path ASes to be subsumed by the stronger case where the entire respective AS is malicious. Excessive requests or data traffic by malicious end hosts within honest on-path ASes can easily be filtered using separate mechanisms.

Monitoring and Enforcement of Reservations. To prevent link flooding attacks, both on the data plane and on the control plane, the N-Tube bandwidth reservation algorithm must run alongside flow-policing mechanisms that effectively detect and block overuse of allocated bandwidth (see, e.g., [28]). On the data plane, we must enforce that bandwidth reservations exist for all traffic and that allocated bandwidth is not overused.

E1 We assume that all data plane traffic has a bandwidth reservation, and that an effective flow-policing mechanism is in place to prevent the overuse of allocated bandwidth by malicious ASes.

The flooding of the bandwidth reservation algorithm itself with reservation requests (which are part of the control plane) can easily be prevented as follows.

E2 We assume that honest ASes limit the frequency of per-AS reservation requests.

Using this mechanism, excessive requests from malicious hosts would not even leave honest ASes and would otherwise be limited by the first honest AS on the path.

Since all traffic must have a valid reservation, N-Tube, by virtue of its properties (G1)–(G5), is capable of preventing link flooding attacks including Coremelt and Crossfire, when run alongside an effective enforcement mechanism satisfying assumptions (E1) and (E2). We will further explain how this is achieved in Section III-E.

III. N-TUBE OVERVIEW

Our N-Tube algorithm enables ASes to reserve bandwidth on network paths by reserving bandwidth on each inter-domain link on the path. A reservation consists of a path, an expiration time, and a bandwidth amount. The reservation is valid for

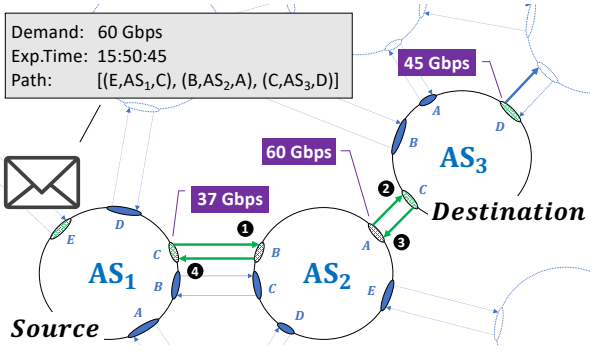

Figure 1: The process of making a reservation

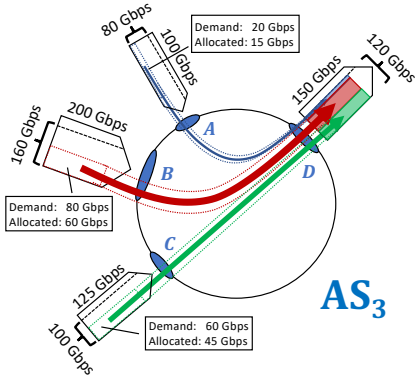

Figure 2: Bandwidth allocation computation at AS_3 distributes the egress link's (adjusted) bandwidth capacity D proportionally to three ingress demands.

a limited time period after which it must be renewed. This allows ASes to probe the network for congestion, and to adjust their reservation paths and demands.

To reserve bandwidth, the source AS chooses loop-free paths to the destination AS (obtained from the control plane, see Section II-B), an amount of bandwidth and an expiration time, combines them in a reservation message, and authenticates it, e.g., with RPKI [29]. Figure 1 illustrates the reservation process. The source AS_1 sends a reservation message demanding `60 Gbps` valid until `15:50:45` on the path given by the list of ASes and their corresponding ingress and egress interfaces $[(E,AS_1,C),(B,AS_2,A),(C,AS_3,D)]$. On the way to AS_3, the reservation message accumulates the amount of bandwidth each AS on the path can allocate on its egress link associated to the path: `37 Gbps`, `60 Gbps`, and `45 Gbps`, respectively. On the return path, each AS allocates the minimum of the accumulated bandwidths, i.e., `37 Gbps`.

N-Tube has two user-specific parameters. First, N-Tube enforces an upper bound, $maxT \in \mathbb{N}$, on how long the expiration time can be set into the future. This forces ASes to update their reservations regularly, roughly every 5 minutes. Second, N-Tube only reserves a fixed portion δ ($0 < \delta < 1$) of each link's total capacity, called the *adjusted capacity*. For any new reservation request, N-Tube initially allocates at most the portion δ of the remaining free capacity, and thereby keeps the rest of the link's capacity available for other new reservations.

A. Bounded Tube Fairness (G5)

The main challenge for a resource allocation algorithm is to treat all reservations *fairly*, and to provide a *lower-bounded* bandwidth allocation for honest ASes, even when adversaries try to congest a link by demanding excessive bandwidth.

To provide *fair* bandwidth allocation, N-Tube bounds excessive demands by the links' capacities, and shares the resulting demands proportionally. This is illustrated in Figure 2 by the bandwidth allocation computation at AS_3 in the above example:

1) AS_3 factors the demands converging at a given egress interface by each ingress interface. These factored demands are called *tubes*, as we visualize them this way.
2) AS_3 bounds the accumulated demand of each tube by its ingress and egress links' adjusted capacities, which we call their *bounded tube demands*.
3) AS_3 proportionally shares the egress link's adjusted capacity between its bounded tube demands.

AS_3 has three interfaces A, B, and C with ingress link capacities `100 Gbps`, `200 Gbps`, and `125 Gbps`, and an interface D with an egress link capacity of `150 Gbps`, respectively. The link's adjusted capacities are obtained by multiplying each link's capacity with $\delta = 0.8$ in this case, and are indicated by the dotted lines. The three demands of `20 Gbps`, `80 Gbps`, and `60 Gbps` from interfaces A, B, and C are factored into three tubes, and the adjusted capacity of `120 Gbps` at interface D is proportionally split among them into `15 Gbps` for A, `60 Gbps` for B, and `45 Gbps` for C, respectively. For example, in case of C this is computed as follows: `45 Gbps` = `60 Gbps` / (`20+80+60 Gbps`) · 0.8 · `150 Gbps`

B. Minimum Bandwidth Guarantee (G4)

By bounding the accumulated demands of tubes in the second step of the bandwidth allocation computation, we guarantee that each tube obtains a fair share of the egress link's capacity. Whenever we must reduce a tube, we say it has *excessive demands*, and we proportionally reduce all demands inside it.

We illustrate how N-Tube computes bandwidth allocations in the presence of adversaries with three examples. We assume that all ASes on the given path are honest, and any AS off this path may be adversarial. The goal of the adversaries is to reduce as much as possible the allocated bandwidth for the honest ASes between interface C and interface D. Hence, we allow adversaries to demand an arbitrary amount of bandwidth to subsequently congest the egress link at interface D. We then show how the bandwidth allocation computation still provides a minimum bandwidth guarantee.

Limit demands on an ingress link by its adjusted capacity: In the Figure 3 example, two adversarial ASes demand in total `400 Gbps` (`150 Gbps` and `250 Gbps`, respectively) of bandwidth through interface B to D. N-Tube bounds these demands by the ingress link's adjusted capacity at interface B of `160 Gbps`. Hence, D's adjusted capacity of `120 Gbps` is split proportionally between `20 Gbps` from A, `60 Gbps` from C, and `160 Gbps`, instead of `400 Gbps`, from B.

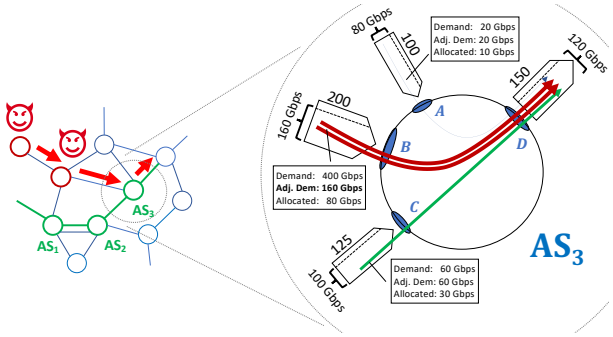

Figure 3: Limit demands on the ingress interface.

Limit each AS's demands by the egress link's capacity: In Figure 4, an adversarial AS demands in total `200 Gbps` to interface D: `80 Gbps` and `120 Gbps` through interfaces A and B, respectively. However, since its combined demand of `200 Gbps` exceeds the egress link's capacity, N-Tube reduces both demands proportionally by a *scaling factor*. The scaling factor is the ratio of the egress link's adjusted capacity to the total adjusted demand of the adversarial AS, i.e., `120/200` = 0.6. This results in the *reduced demands* of `48 Gbps` ($=$ 0.6 $\cdot$ `80 Gbps`) and `72 Gbps` ($=$ 0.6 $\cdot$ `120 Gbps`) from interfaces A and B, respectively. Note that, in this case, each AS must keep per-source AS state, i.e., how much bandwidth each source AS has reserved through this AS. This is feasible since the number of ASes in a network is much smaller than the number of flows. Hence, D's adjusted capacity of `120 Gbps` is split proportionally between `60 Gbps` from interface C, and the reduced demands of `48 Gbps` and `72 Gbps`, from interfaces A and B. The computed allocations are therefore `40 Gbps`, `32 Gbps`, and `48 Gbps`, respectively.

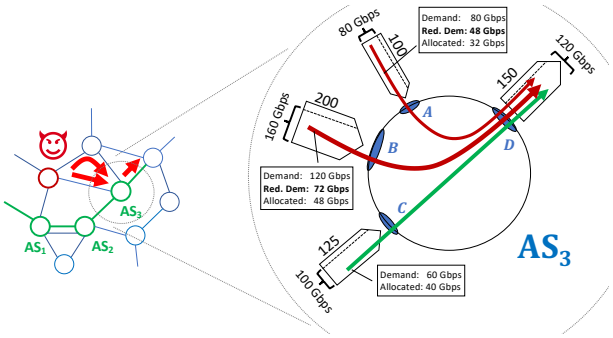

Figure 4: Limit demands on the egress interface.

Worst case, minimum bandwidth guarantees: In Figure 5, all off-path ASes are adversarial, demanding as much as they can on all the ingress links, i.e., a maximum of `80 Gbps` on interface A and `160 Gbps` (`40 Gbps` and `120 Gbps`, respectively) on interface B. This represents a worst-case attack: even when more adversarial ASes are present, their bandwidth demands will be adjusted, and thus limited as described in the two previous examples. The interface D's adjusted capacity of `120 Gbps` is split proportionally between the bounded demands of `80 Gbps` from A and `160 Gbps` from B, and benign demand of `60 Gbps` from C. Hence, this benign demand cannot be reduced to less than `24 Gbps` by any amount of external demands. This provides the minimum bandwidth guarantee for the honest reservation at AS_3.

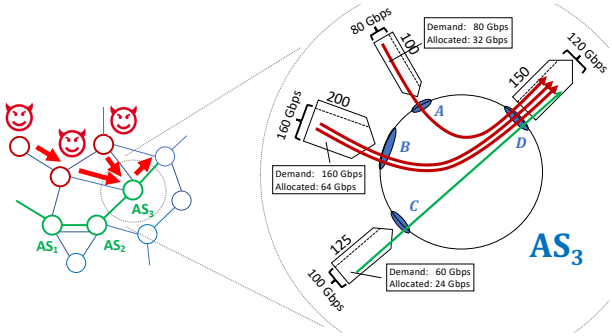

Figure 5: Worst-case minimum bandwidth guarantee.

Global lower bound: By applying the idea from the previous example, we can provide a *local lower bound* llb_3 for the proportion of D's adjusted link capacity that can be allocated to the benign demand. In the worst case, all tubes have excessive demands at interface D. Then llb_3 is given as the ratio of the benign demand of `60 Gbps` and the accumulated adjusted capacities of all ingress links, i.e., $llb_3 = 0.17 \approx$ `60 Gbps/(100+160+80 Gbps)`. Likewise, we can compute llb_1 and llb_2 with respect to AS_1 and AS_2's ingress links' capacities. Note that these local lower bounds do not depends on the adversarial demands.

The *request ratio* $reqRatio_0$ at the source AS_1 is defined as the ratio of the benign demand of `60 Gbps` and the total demand of reservations starting at interface E of AS_1 (see Figure 1). The global lower bound *glb* for the bandwidth that can be allocated to the honest demand is derived as

$$glb = reqRatio_0 \cdot llb_1 \cdot llb_2 \cdot llb_3 \cdot \texttt{120 Gbps}.$$

Note that *glb* only depends on the request ratio at the (honest) source and the capacities of the on-path ASes' ingress links. The request ratio is bounded by the number of reservations the source AS_1 starts at its interface E, i.e., under its own control, and is not influenced by (malicious) reservations (see [27, Appendix C] for details). This is, intuitively, why N-Tube's bandwidth allocation computation provides (G4). Furthermore, the computation splits the adjusted link's capacity proportionally between the non-excessive demands. This illustrates, informally, that N-Tube provides (G5) for each link.

C. Stability (G3)

N-Tube's upper bound *maxT* on the expiration time forces ASes to renew their reservations regularly. This allows N-Tube to stabilize the allocations in a predictable time period *stabT* of constant demands after a burst in demands. The time period *stabT* needed to stabilize demands can be shown to be the

product of *maxT* and the length of the longest reserved path $\hat{p}$ in the network, i.e.,

$$stabT = length(\hat{p}) \cdot maxT.$$

Intuitively, the bandwidth computation at the first AS only depends on requested demand at that AS and the constant bandwidth allocations are then successively propagated to all ASes on the path at each renewal of the reservation. This provides an informal argument that N-Tube achieves the *stability* property (G3).

We will show that, after the entire network stabilizes, N-Tube's bandwidth allocations also satisfy *bounded tube fairness* (G5).

D. Immutability (G2) and Availability (G1)

By reserving only a fraction δ of the available bandwidth, N-Tube can always provide a positive (but possibly small) amount of bandwidth for a new reservation. This ensures *availability* (G1). Unused bandwidth capacities can be used for best-effort traffic. N-Tube does not change established reservations until they either expire or are explicitly deleted by the source (see Section IV). This yields *immutability* (G2).

E. Preventing Link Flooding Attacks

Fairness notions like per-source or per-destination fairness would lead to bandwidth slices of size $\mathcal{O}(1/N)$ in the worst case, where N denotes the number of end hosts (in the billions), i.e., in today's Internet $N \sim 10^{10}$. Even worse, in today's Internet, per-flow fairness allows adversarial hosts to request arbitrarily many flows which squeezes bandwidth slices of honest hosts even further. Specific examples of such an attack are Coremelt [2] and Crossfire [1]. In Coremelt, M bots can reduce bandwidth slices to $\mathcal{O}(1/(M \cdot P))$ by contacting P destination servers. In Crossfire, attacks can reduce bandwidth to $\mathcal{O}(1/M^2)$, where modern botnets contain millions of infected end hosts, i.e., $M \sim 10^7$.

In the context of N-Tube, however, end-hosts are restricted to the bandwidth slices that their edge AS reserves for them. Hence, in attacks like Coremelt and Crossfire, infected end-hosts flooding single Internet links with data traffic are detected by the enforcement mechanism, as described in Section II.B, and are blocked at their edge AS or the next honest neighboring AS. This renders these two attacks ineffective.

A related adaptation of link flooding attacks like Coremelt and Crossfire to the context of N-Tube are colluding malicious ASes that demand excessive amounts of bandwidth to maximally congest a single link in the network. However, this is prevented by the virtue of properties (G1-G5), providing minimum bandwidth guarantees to honest ASes as described in Section III-B. In the worst case, when thousands of malicious ASes, i.e., $n \sim 10^4$, collude to ask for excessive demands on an honest path in the network, bandwidth slices are reduced to $\mathcal{O}(1/n)$. Note that this bound can be further improved when honest source ASes reserve the whole path instead of an intermediate segment (see [27, Appendix C] for details).

In another adaptation of these attacks, malicious ASes flood the control plane with reservation requests hindering honest ASes from making their reservations. However, this can be easily prevented as described in assumption **E2** in Section II-B and is therefore not considered in this work.

IV. Algorithm Details

In this section, we first introduce formal preliminaries and then define the network model, messages, reservation maps, and N-Tube's message processing. We then specify the bandwidth allocation computation and its local properties. Finally, we present our LTS formalization of N-Tube. For full model details, see [27, Appendix D].

A. Notation

Let $\mathbb{1} = \{\bot\}$ denote the unit set, $\mathbb{B} = \{\text{TRUE}, \text{FALSE}\}$ the booleans, $\mathbb{N}$ the natural numbers, and $\mathbb{R}_0^+$ the non-negative real numbers. For $a \leq b$, open and closed intervals are denoted by $]a;b[$ and $[a;b]$.

Given two sets A and B, we denote the *function space* with *domain* A and *co-domain* B by $A \to B$, their *product* by $A \times B$, and their *disjoint sum* by $A + B$. We define partial functions $A \rightharpoonup B = A \to B_\bot$ with $B_\bot = B + \mathbb{1}$, the *support* of a partial function g by $supp(g) = \{a \in A \mid g(a) \neq \bot\}$, and its *range* by $rng(g) = \{b \in B \mid \exists a \in A.\ g(a) = b\}$. We denote partial functions with finite support by $A \rightharpoonup_{fin} B$ and the undefined function with empty support by $\emptyset$. The updated function $f(x \mapsto y)$ is defined by $f(x \mapsto y)(x) = y$ and $f(x \mapsto y)(x') = f(x')$ for all $x' \neq x$. A *record type* is a product type with named projections, e.g., $point = (\!|\ x \in \mathbb{N}; y \in \mathbb{N}\ |\!)$ with elements like $p = (\!|\ x = 1; y = 2\ |\!)$ and fields $p.x$ and $p.y$. The term $p(\!|\ x := 3\ |\!)$ denotes the updated point $(\!|\ x = 3; y = 2\ |\!)$. The inductive set of *lists* over A, denoted by $[A]$, is constructed from the empty list *nil* and the operation $a\#l$, which prepends an element $a \in A$ to a list $l \in [A]$. We write, e.g., $[1,2,3]$[1] for the list $1\#2\#3\#nil$, and $l[n]$ to retrieve the nth element of the list l, counting from 0. For a set A we write $\mathbb{P}(A)$ for its *power set*, $\mathbb{P}_{fin}(A)$ for the set of its finite subsets, and $|A|$ for its *cardinality*.

The functions min and max respectively yield the minimum and maximum element of a non-empty finite set of numbers, and $+\infty$ and 0 for the empty set. We extend them to tuples, lists, and records of numbers (by taking the set of their components), and to partial functions with finite support and numerical co-domain (by taking the range).

B. Network and Messages

Network. We model the *network* as a weighted, directed graph (N, E, cap), for which we give a simplified definition:

- The nodes N are given by the set $V \times I$, where V is a finite set of vertices (ASes), and I provides a set of identifiers for interfaces inside of each AS.
- The finite set of directed edges $E \subseteq N \times N$ represents the physical *links* between ASes. Given a link $((u,e),(v,i)) \in E$, we call e its *egress* interface at AS u and i its *ingress*

[1] Note the syntactic difference between the closed interval $[1;3]$, the pair $(1,3)$, and the two-element list $[1,3]$.

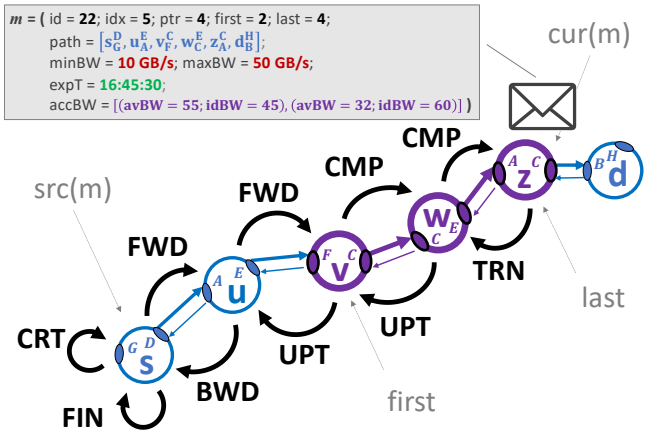

Figure 6: The message m is processed along p by the events described in Section IV-E2.

interface at AS v, respectively. We assume that at any AS interface there is either exactly one *ingress* and one *egress* link or no link at all.

- The *capacity* of each link is given by the non-negative real-valued function $cap : E \to \mathbb{R}_0^+$. Since a link $l = ((u,e),(v,i))$ is uniquely defined by (u,e) (and (v,i)), we identify $cap(l)$ with $cap(u,e)$ (and $cap(v,i)$).

We define the type of *paths* $\mathscr{P}$ as lists of records with ingress interface *inI*, AS identifier *as*, and egress interface *egI*:

$$\mathscr{P} = \Big[\!\!\Big(\, inI \in I; as \in V; egI \in I \,\Big)\!\!\Big].$$

Given a network, we call a path $p \in \mathscr{P}$ *valid*, if (i) it is non-empty, (ii) each edge corresponding to p's ingress and egress interfaces is an inter-AS link, i.e., it starts and ends in distinct ASes, and (iii) its set of links is *connected* and *directed*, i.e., the edges connect the ASes in p, and they all point in the same direction, and (iv) it is *loop-free*, i.e., each AS occurs at most once on the path.

In what follows, we denote the elements of sets V and I with lowercase letters: for V we use the letters s, x, v, and z, for ingress interfaces i and i', and for egress interfaces e and e'. We also write the ingress interface as subscripts and the egress interface as superscripts to the AS identifier, e.g., x_i^e.

Messages. There are two types of messages: *reservation* and *deletion messages*. The message fields identify the reservation, state how the message should be routed through the network, how much bandwidth should be reserved, and until when the reservation should be valid. We introduce the fields of a reservation message m in Figure 6:

- The source $s \in V$ of the path (see below) can choose any reservation ID $id \in \mathbb{N}$ (= 22). The pair $(s,22)$ of source identifier and reservation ID uniquely determines a reservation in the network. Furthermore, the source provides an index idx (= 5) indicating which version of the reservation the messages refers to. Version indices are used to update existing reservations (explained later).
- The field $path \in \mathscr{P}$ (= p) provides the path, and the field $ptr \in \mathbb{N}$ (= 4) provides the pointer to the AS ($p[4].as = z$) where the message is currently processed. The fields $first \in \mathbb{N}$ (= 2) and $last \in \mathbb{N}$ (= 4) refer to the first and last AS on p, respectively.
- The minimum $minBW \in \mathbb{R}_0^+$ (= 10 GB/sec) and maximum bandwidth $maxBW \in \mathbb{R}_0^+$ (= 50 GB/sec) state the range of bandwidth the source AS would like to reserve. Hereby, $maxBW$ states the source's *demand* in the reservation request. In case the source's demand cannot be provided on p, $minBW$ states the *minimal* amount of bandwidth the source is willing to accept as a reservation.
- The expiration time $expT \in \mathbb{N}$ (= 16:45:30) indicates when the reservation expires and must be deleted by the ASes on the path p.
- The list of bandwidth values $accBW \in [(\!|\, avBW, idBW \in \mathbb{R}_0^+ \,|\!)]$ indicates the *available* and *ideal* bandwidth the previous ASes (v and w in the example) were able to provide, as explained in Section IV-D.

The functions *src*, *first*, *cur*, and *sgmt* on valid messages extract from *m.path* the *source AS*, the *first AS*, the *current* AS with its ingress and egress interfaces, and the *set of ASes between first and last* (including the endpoints), respectively. In the example of Figure 6 $src(m) = s$, $first(m) = v$, $cur(m) = z_A^C$, and $sgmt(m) = \{v,w,z\}$. In a deletion message, the fields *first*, *last*, *minBW*, *maxBW*, *expT*, and *accBW* are omitted.

The type of *messages* $\mathscr{M} = \mathscr{M}_R + \mathscr{M}_D$ is the disjoint sum of *reservation messages* $\mathscr{M}_R$ and *deletion messages* $\mathscr{M}_D$. A message m is *valid*, if *m.path* is a valid path, and for its pointers it holds that $0 \le ptr, first, last \le length(m.path)$ and $first < last$. The bandwidth range must be a non-empty interval, i.e., $0 \le m.minBW \le m.maxBW$ and $m.maxBW > 0$, and thus only non-zero bandwidth allocations are allowed.

C. Message Processing

Reservation Maps. Each AS maintains its own reservation map where all currently valid reservations with a path traversing this AS are stored. A reservation map is a partial function that maps a source and a reservation ID to a record containing the following fields: the reservation's *path*, the pointers *ptr*, *first* and *last*, and a *version map vrs*. For example, in the reservation map $resM_z$ of AS z, the entry rs corresponding to message m from Figure 6 is

$$resM_z(s,22) = \\ (\!|\, path = p; ptr = 4; first = 2; last = 4; vrs = verM \,|\!).$$

N-Tube allows source ASes to flexibly update their reservations multiple times before they expire, and therefore stores different versions of each reservation in the *version map vrs*.

A version map is a partial function that maps each reservation index to a record containing the following fields: the minimal bandwidth $minBW$, the maximal bandwidth $maxBW$, the ideal bandwidth $idBW$ computed by the previous AS on p, the expiration time $expT$ given by m, and the reserved bandwidth

resBW determined by N-Tube. We call the reservation's entries in the version map its *versions*, e.g., for m

$$verM(5) = \{\!|\ minBW = 10\text{ GB/s}; maxBW = 50\text{ GB/s};\\ idBW = 60\text{ GB/s}; resBW = 32\text{ GB/s};\\ expT = 16{:}45{:}30\ |\!\}.$$

A version vr is *currently valid* at time t, written $cvalid(vr,t)$, if it is *successful*, i.e., $vr.minBW \leq vr.resBW$ and *not expired*, i.e., $vr.expT \geq t$. The reservation's bandwidth *demand* and *allocation*, *demBW* and *allocBW*, are defined as the maximum of its currently valid versions' *maxBW* and *resBW*, respectively.

$$demBW(rs,t) = \max_{vr \in rng(rs.vrs)} \{vr.maxBW \mid cvalid(vr,t)\}$$
$$allocBW(rs,t) = \max_{vr \in rng(rs.vrs)} \{vr.resBW \mid cvalid(vr,t)\}$$

The maximum is taken as the source can send traffic using any existing version of its reservations. In this way, sufficient bandwidth is guaranteed to be available in the worst case.

Reservation Process. N-Tube processes a reservation message depending on its direction and position on the path. As shown in Figure 6, suppose that AS s intends to make a new reservation *id* on a path p. It then creates (CRT) a reservation message m containing p in its *path* field. The ASes located before *first* on p just forward (FWD) the message along p by increasing m's *ptr* field. If m reaches the ASes between *first* and *last*, each AS x computes (CMP) m by:

1) checking that $resM_x$ does not contain a reservation at (s,id), or there is a reservation at (s,id) for the same path with no valid version map entry at *idx*,
2) computing how much bandwidth is *available* at x and how much x can *ideally* provide for the reservation (see Section IV-D for the details of the computation),
3) updating m to a new message m' by appending the computed results to *accBW* and by incrementing *ptr*,
4) sending m' to the next AS on the path p, and
5) adding a new version at index *idx* of the reservation identified by (s,id) in $resM_x$.

After the last AS z (indicated by pointer *last*) on the path has processed m, it returns (TRN) the message m'. During the backward traversal, each AS on the path extracts how much bandwidth *finBW* could be reserved on the entire path by taking the minimum of *maxBW* and *accBW*, i.e., what have been computed in the forward traversal

$$finBW(m) = \min\left(m.maxBW, \min(m.accBW)\right).$$

Analogously to the forward traversal updates, an AS updates (UPT) its reservation map according to the same two cases: (i) the reservation was successful, i.e., $m.minBW \leq finBW$, and each AS on the path updates the reserved bandwidth of the corresponding version in its reservation map to *finBW*, or (ii) there was not enough bandwidth available on the path, i.e., $m.minBW > finBW$, and each AS deletes the corresponding version from its reservation map. The ASes between *first* and source AS s simply send the message backwards (BWD) without processing it until s finally receives it (FIN).

Renewal and Deletion. If s intends to *renew* one of its reservations, it sends a new reservation message m, containing an updated bandwidth range and expiration time, along the previous path p. To delete a reservation's version, a source AS sends a *deletion* message along the corresponding path.

D. Fair Bandwidth Allocation

The heart of the N-Tube algorithm is its bandwidth allocation computation. We assume that a *valid* reservation message m was sent by its source AS s and arrives at an AS x lying between *first* and *last* on m's path at the current time t. First, N-Tube derives its source s ($= src(m)$) and its current AS, ingress, and egress interfaces x_i^e ($= cur(m)$). Given m and $resM_x$, the bandwidth allocation computation determines:

- the *available* bandwidth, i.e., how much bandwidth remains on the link at the egress interface e, and
- the *ideal* bandwidth, i.e., how much bandwidth is allocated to s with respect to all active reservations in $resM_x$ between interfaces i and e.

The corresponding functions *avail* and *ideal* are defined below. To simplify notation, we fix the message m and its elements s, *id*, x, i, and e, and omit $resM_x$, t, and the parameter δ as arguments. The functions *resSr*, *resEg*, and *resIn* extract a reservation's source AS and the current AS' egress and ingress interfaces, respectively. For full details, see [27, Appendix D5].

1) Available Bandwidth Computation: Given the message m, the function *avail* computes how much bandwidth is *available* on the link at the egress interface e of AS x

$$avail(e) = \ \delta \cdot \left(cap(x,e) - \sum_{\substack{r' \in rng(resM_x):\\ resEg(r')=e}} allocBW(r') \right).$$

It subtracts the aggregated *allocated bandwidth* of all currently valid reservations with egress interface e from the link's total capacity $cap(x,e)$, and multiplies the result with the parameter $0 < \delta < 1$. The factor δ guarantees available bandwidth for subsequent reservations.

2) Limiting Excessive Demands: To avoid that s reserves more bandwidth in one request than physically available, N-Tube limits the bandwidth demand $demBW(r)$ of a reservation r by the ingress and egress links' adjusted capacities. The resulting *requested demand* of a reservation r is defined by

$$reqDem(r) = \min\{\delta cap(x,i), \delta cap(x,e), demBW(r)\}.$$

As illustrated in Section III, a source's aggregated demands at a given link may exceed the link's capacity, even if none of its individual requests does. We now formally define the notion of a source having excessive demands on a link, and of an adjusted version of the requested demand, *adjReqDem*, to account for such demands.

The *egress demand* of s on e is defined as the aggregate over its *requested demands* with e

$$egDem(s,e) = \sum_{\substack{r' \in rng(resM_x):\\ resSr(r')=s\\ resEg(r')=e}} reqDem(r').$$

We analogously define the *ingress demand* on interface i.

Definition 1 (Excessive Demands)**.** We say an AS s *has excessive demands on the egress link* e, if $egDem(s,e) > \delta cap(x,e)$. Otherwise, we say s *has moderate demands on* e. We call an egress link e *congested* if $egDem(s',e) > cap(x,e)$. Analogous definitions apply to ingress links.

To account for the case where s has excessive demands on the egress link e, we adjust the requested demand of a reservation r by multiplying it with the minimum of the corresponding ingress and egress *scaling factors*, yielding the *adjusted requested demand*:

$$adjReqDem(r) = \min\{inScalFctr(s,i), egScalFctr(s,e)\} \cdot reqDem(r,i,e).$$

with s, i, and e the corresponding source AS, ingress and egress interface of r, respectively. We compute for source AS s the *egress scaling factor* on the egress link e as the source's proportion of the total *egress demand* bounded by the egress link's capacity, given by

$$egScalFctr(s,e) = \frac{\min\{\delta cap(x,e), egDem(s,e)\}}{egDem(s,e)}.$$

We analogously define the source's *ingress scaling factor*.

3) Ideal Bandwidth Computation: Given a message m with x^i_e on its path, the function *ideal* computes how the adjusted capacity $\delta \cdot cap(x,e)$ of the egress link e is shared in a so-called *bounded tube fair* manner among all existing reservations (in $resM_x$) with the same egress link e:

$$ideal(s,id,i,e) = reqRatio(s,id,i,e) \cdot tubeRatio(i,e) \cdot \delta \cdot cap(x,e).$$

This (1) proportionally splits the egress link's adjusted capacity between all ingress links by multiplying it with *tubeRatio*, and (2) further splits the result proportionally between all reservations from i to e by multiplying it with *reqRatio*.

We define these two ratios in the following.

Tube Ratio: The *tube ratio* between an ingress interface i and an egress interface e is computed as the ratio of the *bounded tube demand* between i and e, given by $\min\{cap(x,i), tubeDem(i,e)\}$, and the aggregated bounded tube demands at e

$$tubeRatio(i,e) = \frac{\min\{\delta cap(x,i), tubeDem(i,e)\}}{\sum_{i' \in I} \min\{\delta cap(x,i'), tubeDem(i',e)\}}.$$

Taking the minimum with respect to the corresponding ingress link's capacity guarantees that its respective portion of the tube demand compared to the other ingress links' tube demands is always bounded. This prevents the reserved bandwidth for other ingress links from being reduced ad infinitum.

The *tube demand* between an ingress interface i and an egress interface e aggregates their *adjusted requested demands*

$$tubeDem(i,e) = \sum_{\substack{r' \in rng(resM_x): \\ resIn(r')=i \\ resEg(r')=e}} adjReqDem(r').$$

Request Ratio: The *request ratio* of a reservation (s,id) between i and e is the ratio between its *adjusted ideal bandwidth demand* (provided by the predecessor on the reservation's path) and the *transit demand* at i:

$$reqRatio(s,id,i,e) = \frac{adjIdDem(s,id,i,e)}{transitDem(i)}.$$

The function *adjIdDem* is defined similarly to *adjReqDem* but for the previously computed ideal bandwidth, and *transitDem* is the aggregation of all *adjIdDem*'s at ingress interface i.

E. Formalizing N-Tube

We formalize N-Tube using labeled transition systems, as this is a widely known model that is well-suited for hand-written proofs. For our statistical analysis of N-Tube, we will transform these models into probabilistic rewrite systems and analyze them using Maude [26] (Section VI).

A *labeled transition system* (LTS) $\mathfrak{T} = (\Sigma, \Sigma_0, \Lambda, \Delta)$ consists of a state space Σ, a set of initial states $\Sigma_0 \subseteq \Sigma$, a set of labels Λ, also called *events*, and a (labeled) transition relation $\Delta \in \Lambda \to \mathbb{P}(\Sigma \times \Sigma)$. *Executions* of $\mathfrak{T}$ are functions of type $\mathfrak{E} = \mathbb{N} \to \Sigma \times \Lambda$ such that any $\pi = \{(\sigma_n, \lambda_n)\}_{n \in \mathbb{N}} \in \mathfrak{E}$ starts in an initial state, i.e., $\sigma_0 \in \Sigma_0$, and progresses according to the transition relation Δ, i.e., for all $n \in \mathbb{N}$, $(\sigma_n, \sigma_{n+1}) \in \Delta(\lambda_n)$.

To specify concrete models, we often use Λ-indexed families of *guards* $G_\lambda : \Sigma \to \mathbb{B}$ and *update* functions $U_\lambda : \Sigma \to \Sigma$. The induced transition relation is

$$\Delta(\lambda) = \{(\sigma, \sigma') \mid G_\lambda(\sigma) \wedge \sigma' = U_\lambda(\sigma)\}.$$

The relation $\sigma' = U_\lambda(\sigma)$ is called the *action* of the event. For example, in the domain of banking, an event to withdraw an amount a of money from an account is specified by $withdraw(a) = \{(\sigma, \sigma') \mid \sigma.bal \geq a \wedge \sigma'.bal = \sigma.bal - a\}$. In this case, any state (record) field f that is not updated is implicitly left unchanged, e.g., $\sigma'.f = \sigma.f$.

We fix the environment: a network graph (N, E, cap) as in Section IV, a partition $V = H + M$ (with H and M the sets of honest and malicious ASes), and the fraction $0 < \delta < 1$ of the link's adjusted capacity. We model the behaviors of both honest and malicious ASes (see [27, Appendix D] for full details).

1) States: We define the set of *states* Σ as the record

$$\Sigma = (\!| \; time \in \mathbb{N}; buf \in Buff; res \in ResMap; kwl \in \mathbb{P}(\mathscr{M}) \; |\!).$$

A *state* $\sigma \in \Sigma$ describes a snapshot of the system at a given point in time, denoted by its *time* field. We assume discrete time is *loosely* synchronized between all ASes, i.e., compared to the minimal duration of reservations (on the order of minutes), the discrepancy of time measurements between AS (on the order of 100 ms) is negligible (cf. Assumption **N3**).

The field *buf* of type $Buff = V \times I \to \mathbb{P}_{fin}(\mathscr{M})$ models network *buffers*, where $buf(x,i)$ holds the set of messages arrived at interface $i \in I$ of AS $x \in V$. The field *res* models all ASes' *reservation maps* as presented in Section IV-C. Finally, the field *kwl* models the *attackers' knowledge*: the set of messages created, collected, and shared by all malicious ASes. This models the attackers' collusion (cf. Assumption **A1**).

2) Events: The set of events Λ consists of system events and environment events. System events formalize the N-Tube algorithm: its *message processing* events and an internal event that *removes expired reservations* in each AS. There are different message processing events depending on a message's type, its location on the path, and the direction of the path traversal (cf. Figure 6). This results in seven events describing reservation message processing, three handling deletion messages, and one for dropping messages. Three events model the environment: a *time progress* (clock tick) event and two events modeling the *attackers' capabilities*. Below, we present the attacker events and one representative message processing event.

Attacker Events. Malicious ASes can execute two events: (i) receive a message, partially modify it, and store the resulting message in the attackers' knowledge *kwl*; (ii) send a message in *kwl* to any neighbor AS in the network. Recall that *kwl* also includes any message in $\mathscr{M}$ with a malicious source AS. We present these two events in turn. The collect event (i) is defined by

$$\begin{aligned} &CLT(m,m' \in \mathscr{M}, a \in M, i \in I) = \{(\sigma,\sigma') \mid \\ &\quad \texttt{- guards -} \\ &\quad m \in \sigma.buf(a,i) \ \wedge \ m' \approx m \ \wedge \\ &\quad \texttt{- actions -} \\ &\quad \sigma'.kwl = \sigma.kwl \cup \{m'\} \ \}. \end{aligned}$$

Here, an attacker $a \in M$ receives a message m from his buffer $buf(a,i)$ at interface i, possibly modifies it, and adds the resulting message to *kwl*. The equivalence relation $m \approx m'$ expresses that m and m' coincide except on their mutable fields *ptr* and *accBW*. This prevents off-path attackers from spoofing reservation requests from other ASes. This event models Assumption **A2** (i-ii). In an N-Tube implementation, the source AS would sign the immutable fields with its private key, while the mutable fields would remain unprotected.

In the attack event (ii), an attacker a can send any message m in *kwl* to any neighbor AS v by adding m to v's buffer $buf(v,i)$. This event models Assumption **A2** (iii).

$$\begin{aligned} &ATK(m \in \mathscr{M}, a \in M, v \in H, i,e \in I, t \in \mathbb{N}) = \{(\sigma,\sigma') \mid \\ &\quad \texttt{- guards -} \\ &\quad m \in \sigma.kwl \ \wedge \ ((a,e),(v,i)) \in E \ \wedge \\ &\quad \texttt{- actions -} \\ &\quad \sigma'.buf = \sigma.buf((v,i) \mapsto \sigma.buf(v,i) \cup \{m\} \ \}. \end{aligned}$$

These two events model powerful attack capabilities. Malicious ASes can anytime make arbitrary reservation requests from their own ASes, partially modify observed requests, replay old messages, and collude through out-of-band channels to share their knowledge and synchronize attacks. However, attackers cannot spoof messages from honest ASes, modify reservations stored in the reservation maps of honest ASes, or change the system's global time.

Message Processing Events. Here we show the definition of the compute event, which is the most representative message processing event:

$$\begin{aligned} &CMP(m,m' \in \mathscr{M}_R, v \in H, i \in I, t \in \mathbb{N}) = \{(\sigma,\sigma') \mid \\ &\quad \texttt{- guards -} \\ &\texttt{(1)} \ m \in \sigma.buf(v,i) \ \wedge \ \texttt{(2)} \ \sigma.time = t \ \wedge \\ &\texttt{(3)} \ PathCheck(m.path) \ \wedge \texttt{(4)} \ ResMsgCheck(m,\sigma.time) \ \wedge \\ &\texttt{(5)} \ ResMapCheck(\sigma.res,m,v) \ \wedge \texttt{(6)} \ m.first \leq m.ptr < m.last \ \wedge \\ &\texttt{(7)} \ m.path[m.ptr].inI = i \ \wedge \texttt{(8)} \ m' = compute(m,\sigma.res) \ \wedge \\ &\quad \texttt{- actions -} \\ &\quad \sigma'.res = save(v,\sigma.res,m') \ \wedge \\ &\quad \sigma'.buf = forward(v,i,\sigma.buf,m,m') \ \ . \end{aligned}$$

Upon receiving a reservation message m at interface i (first guard) at time t (second guard), AS v allocates bandwidth using the function *save* (first action), and forwards the modified reservation message m' using the function *forward* (second action). All unmentioned fields remain unchanged. Guards (3–5) ensure that m is well-formed and compatible with existing reservations in v's reservation map that corresponds to m. Guard (6) determines whether v is on the path segment, i.e., m's pointer is between *first* and *last*. Guard (7) checks if m traverses the path in the forward direction, i.e., if the arrival interface i of AS v matches the corresponding ingress interface given on m's path field. The last guard models the computation of the modified message m', using the function *compute* to update of received message m's *accBW* field.

$$\begin{aligned} &compute(m \in \mathscr{M}_R, resM \in ResMap) = \\ &\quad \textbf{let} \ newBW = (\!| \ avBW := avail(m,resM); \\ &\qquad\qquad\qquad idBW := ideal(m,resM) \ |\!) \\ &\quad \textbf{in} \ m(\!| \ accBW := newBW \ \# \ m.accBW \ |\!). \end{aligned}$$

This function determines the available and ideal bandwidths that AS v can allocate using the functions *avail* and *ideal* from Section IV. The results are appended to m's *accBW* field.

V. Properties

In this section, we first define the notions of valid executions, successful reservations, and constant demands, which are used to specify N-Tube's global properties (G1–G5). For the sake of readability, we give here a semi-formal versions of these definitions and we refer the reader to [27, Appendix E] for the full formal details and proofs.

Definition 2 (Valid Executions)**.** An execution π is valid if (i) time grows unboundedly on π and (ii) all messages in the buffers of honest ASes are processed in at most time *bufT*.

These assumptions are satisfied if all honest ASes run a fair scheduling algorithm (e.g., round-robin) to prevent message starvation and messages are dropped in case of buffer overflow. We will express N-Tube's properties as predicates over valid executions $\pi = \{(\sigma_n,\lambda_n)\}_{n\in\mathbb{N}}$.

Properties (G1) and (G2) assume that a *successful* reservation has been established by an honest source AS.

Table I: Formalizing global properties (G1–G5).

Property	Formula
(G1) Availability: If an honest AS s makes a successful reservation m at time t, then some non-zero bandwidth will be reserved on its path until it expires.	$\forall m \in \mathscr{M}_R, s \in V, t \in \mathbb{N}, n \in \mathbb{N}, v \in sgmt(m).$ $Succ(s,m,t) \wedge \sigma_n.time \in]t; m.expT]$ $\Rightarrow \sigma_n.res_v(s,m.id).vrs(m.idx).resBW > 0$
(G2) Immutability: If an honest AS s makes a successful reservation m at time t, the reserved bandwidth stays the same for all ASes on its path until it expires.	$\forall m \in \mathscr{M}_R, s \in V, t \in \mathbb{N}, n, n' \in \mathbb{N}, v, v' \in sgmt(m).$ $Succ(s,m,t) \wedge \sigma_n.time, \sigma_{n'}.time \in]t; m.expT]$ $\Rightarrow \sigma_n.res_v(s,m.id).vrs(m.idx).resBW = \sigma_{n'}.res_{v'}(s,m.id).vrs(m.idx).resBW$
(G3) Stability: If there are constant demands D between t_0 and t_1, then all reservations allocate the same amount of bandwidth from $t_0 + stabT$ until t_1.	$\forall D \in \mathscr{D}, t_0, t_1 \in \mathbb{N}, n, n' \in \mathbb{N}, r, r' \in Res, v \in H, m \in rng(D).$ $Stab(D,t_0,t_1) \wedge \sigma_n.time, \sigma_{n'}.time \in]t_0 + stabT; t_1] \wedge$ $r = \sigma_n.res_v(src(m),m.id) \wedge r' = \sigma_{n'}.res_v(src(m),m.id)$ $\Rightarrow allocBW(r, \sigma_n.time) = allocBW(r', \sigma_{n'}.time)$
(G4) Minimum Bandwidth Guarantee: For constant demands D between t_0 and t_1 and for any honest AS's successful reservation, there is a lower bound on the allocated bandwidth that only depends on the request ratio on the first link, a factor G depending on the path's link capacities, and $m.maxBW$.	$\forall D \in \mathscr{D}, t_0, t_1 \in \mathbb{N}.\ Stab(D,t_0,t_1)$ $\Rightarrow \exists \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time = t_0 + stabT$ $\wedge\ \forall m \in rng(D), s, f \in nodes(m).\ s = src(m) \wedge f = first(m) \wedge Succ(s,m,t_0)$ $\Rightarrow\ \exists\ G > 0.\ \forall n > \tilde{n}, v \in sgmt(m).\ \sigma_n.time \in]t_0 + stabT; t_1]$ $\Rightarrow\ allocBW(\sigma_n.res_v(s,m.id), \sigma_n.time) \geq G \cdot reqRatio(m, \sigma_{\tilde{n}}.res_f) \cdot m.maxBW$
(G5) Bounded Tube Fairness: For constant demands D between t_0 and t_1, in the absence of congestion, the bandwidth of egress links is allocated proportionally between tube demands and, in case where tube demands exceed their ingress links' capacities, their tube ratio is bounded.	$\forall D \in \mathscr{D}, t_0, t_1 \in \mathbb{N}.\ Stab(D,t_0,t_1)$ $\Rightarrow \exists \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time = t_0 + stabT$ $\wedge\ \forall m \in rng(D), v \in sgmt(m) \cap H, i, i', e \in I, n > \tilde{n}.$ $tubeDem_v(i,e) \in]0; \delta cap(v,i)] \wedge tubeDem_v(i',e) \in]0; \delta cap(v,i')]$ $\Rightarrow \frac{tubeRatio_v(i,e)}{tubeRatio_v(i',e)} = \frac{tubeDem_v(i,e)}{tubeDem_v(i',e)}$

Definition 3 (Successful Reservation)**.** We say *an honest source $s \in H$ makes a successful reservation confirmed by the message $m \in \mathscr{M}_R$ at time t*, written $Succ(s,m,t)$, if the following three conditions hold: (i) m's path only contains honest ASes; (ii) the source s confirms m at time t with sufficient bandwidth, i.e., there exist $n \in \mathbb{N}$ and $i \in I$ such that $\lambda_n = FIN(m,s,i,t)$ and $finBW(m) \geq m.minBW$; and (iii) There is no deletion event matching the reservation $(src(m), m.id)$ and version $m.idx$ before m's expiration time.

For properties (G3–G5) we model "constant bandwidth demands" as a partial function $D \in \mathscr{D}$ with $\mathscr{D} = V \times \mathbb{N} \rightharpoonup_{fin} \mathscr{M}_R$ such that $D(s,id) = m$ implies $src(m) = s$ and $m.id = id$. We say that a reservation message m *corresponds to* D if $(src(m), m.id) \in supp(D)$ and m coincides with $D(src(m), m.id)$ on all fields except ptr, $expT$, $minBW$, and $accBW$. The *stabilization time*

$$stabT = \max\{length(m.path) \mid m \in rng(D)\} \cdot maxT$$

is the maximal time that the reservation requests in $rng(D)$ must be renewed along their paths to reach a stable state.

Definition 4 (Constant Demands)**.** An execution $\pi \in \mathfrak{E}$ *has constant demands $D \in \mathscr{D}$ between t_0 and $t_1 \geq t_0 + stabT$*, written $Stab(D,t_0,t_1)$, if (i) for all $(s,id) \in supp(D)$, the source AS s has successfully made a reservation confirmed by a message m corresponding to $D(s,id)$ before time t_0, and successfully renews this reservation without any gaps until t_1; (ii) any reservation confirmed by a reservation message m between t_0 and t_1 corresponds to D; and (iii) there are no deletion events between t_0 and t_1 for reservations given by $supp(D)$.

Table I shows our formal specification of properties (G1–G5) under valid executions. Note that these properties hold for reservations along honest paths (cf. Assumption **A3**).

Theorem 1. Our LTS model of N-Tube satisfies properties (G1–G5).

The inductive proofs are given in [27, Appendix E].

VI. Statistical Analysis of N-Tube

Our qualitative analysis of N-Tube by inductive proofs (Section V) establishes the desired correctness and security guarantees, but it offers no insight into the actual dynamics of these guarantees. We therefore additionally conduct quantitative measurements about these guarantees. In particular, we use Maude-based simulation and statistical model checking (SMC) to analyze N-Tube with respect to properties (G1–G5).

Our goal is twofold: (i) to validate our mathematical proofs via independent machine-checked statistical verification; and (ii) to explore quantitative aspects of N-Tube in various adversarial scenarios, with respect to stability, fairness, and resistance to malicious power, using statistical estimations, which goes beyond the inductive proofs.

A. Why Maude and SMC?

Quantitative system analysis typically requires an executable artifact. As rewriting logic [30] is a generic framework for specifying the semantics of a wide range of computation models, LTSs can be naturally expressed as *rewrite theories* in it, and executed as *system modules* in Maude [26]. A rewrite theory consists of an equational theory, specifying the system's data types, and a collection of *labeled conditional rewrite*

rules of the form `crl [l] : t => t' if cond`, where l is a label. Such a rule specifies a transition from a system state, represented by the term t, to a new state t', provided the condition *cond* holds.

The Maude system supports machine-checkable and automated formal analysis, including simulation and SMC [26], [23]. In particular, compared to conventional emulations, SMC can verify a property specified, e.g., in a *stochastic temporal logic*, up to a *statistical confidence level* by running Monte-Carlo simulations of the system model. The expected value $\bar{v}$ of a property query belongs to the interval $[\bar{v}-\frac{\beta}{2}, \bar{v}+\frac{\beta}{2}]$ with $(1-\alpha)$ statistical confidence, where parameters α and β determine when an SMC analysis stops performing simulations [23].

The Maude ecosystem has been very successful in analyzing high-level designs of a wide range of distributed and networked systems [31], [32], [33], [34], [35], [36]. In particular, Maude-based validation using SMC provides additional confidence about claimed statements by analyzing large parameter spaces. Maude-based SMC performance predictions have also shown good correspondence with implementation-based evaluations under realistic deployments [24], [25].

B. Model Transformation

We first express the N-Tube LTS model from Section IV-E as an equivalent untimed, nondeterministic rewrite theory. For the statistical analysis, we then transform this rewrite theory into a *timed*, *purely probabilistic* rewrite theory, following the methodology in [37]. In particular, the transformation assigns to each message a delay sampled from a *continuous probability distribution*, which determines the firing of the rule receiving the message. The resulting model is free from unquantified nondeterminism (in that all transitions are associated with probabilities) and can be simulated by the original model.

The system state of the transformed model consists of a *multiset* of *objects*, including a *scheduler* object maintaining the global clock, and *messages*. An object of class C is represented as a term $< o : C \mid att_1 : val_1, \ldots, att_n : val_n >$, with o the object's identifier, and val_1 to val_n the current values of attributes att_1 to att_n. An incoming message of the form `{t, msg}` is ready to be consumed at the global time t, while an outgoing message of the form `[t+d, msg]` will be delivered in d time units after t where the message delay d is sampled from some continuous probability distribution. Each message *msg* has the form `to o from o' : mp`, with o, o', and *mp* the message receiver, sender, and payload, respectively. The scheduler object is specified to advance the global time and to deliver outgoing messages at the specified times.

We specify N-Tube's dynamic behaviors in Maude by translating its events into rewriting rules. Consider the message processing event *CMP* in Section IV-E2. The following transformed conditional rule `[cmp]` specifies that, upon receiving a reservation message `res(M)` at global time `T` (line 2), the AS `O` updates its local reservation map accordingly (using the `save` function; line 7), and forwards the modified message (determined by the `compute` function) to `next` hop (line 9):

```
1  crl [cmp] :
2  {T,  to O from O' : res(M)}
3  < G : Table | links : LS, ATS' >
4  < O : As | resMap : RM, ATS >
5  =>
6  < G : Table | links : LS, ATS' >
7  < O : As | resMap : save(M,O,RM,LS,AVL,IDL), ATS >
8  [T + lognormal(μ,σ),
9     to next(O,M) from O : compute(M,AVL,IDL)]
10  if (atSrt(M) or onPth(M)) /\ pathCheck(M)
11     /\ resMsgCheck(M,T) /\ resMapCheck(M,RM)
12     /\ AVL := avail(LS,O,RM,T,M)
13     /\ IDL := ideal(LS,O,RM,T,M) .
```

where the network topology and all `links`' capacities are stored in a global "table" `G` (lines 3 and 6). The message delay is probabilistically sampled from the *lognormal* distribution (to mimic the real-work network environment [38]), parametric on the mean μ and standard deviation σ, each time this rule applies (line 8). The functions `avail` (line 12), `ideal` (line 13), `save`, and `compute`, as well as the predicates in the condition (lines 10 and 11), are defined following Section IV-E. The variables `ATS` and `ATS'` refer to the rest of attributes that do not affect the next state.

C. Statistical Analysis

We investigate the following questions about N-Tube using our statistical analysis:

- Are the statistical verification results consistent with our hand-written inductive proofs of (G1–G5)?
- How does N-Tube actually perform in worst- and average-case malicious scenarios with respect to stability and fairness? In particular, how does it resist increasing attack power such as total malicious demands?

1) Benchmark: To statistically analyze N-Tube's properties we implement three *parametric* generators: a topology generator (TG), a path generator (PG), and a workload generator (WG). We use these to probabilistically generate a different initial state for each simulation in an SMC analysis. Specifically, TG generates *scale-free* Internet topologies with strongly connected ASes, which is also characteristic of the realistic AS-level Internet graph in the CAIDA benchmark for Internet data analysis [39]. Each link between nodes is assigned a bandwidth probabilistically sampled from an interval. PG then explores the generated graph, and collects paths from *sources* to *destinations*. WG provides the generated sources, including adversaries, with reservations, renewals, and deletions on a probabilistic basis, where each of these three types of requests is parametric in the algorithm-specific parameters (such as *maxBW* and *expT*). See Appendix B for a complete list of the generators' 18 parameters and their default values.

2) Experimental Setup: We employed a cluster of 50 d430 Emulab machines [40], each with two 2.4 GHz 64-bit 8-Core E5-2630 processors, to parallelize SMC with the PVeStA tool [41] (part of the Maude ecosystem). We set the statistical confidence level to 95%, i.e., $\alpha = 0.05$, and the size parameter β to 0.01 for all our experiments.

3) Analysis Results: We have subjected the transformed Maude model to the above generators and PVeStA, and performed three sets of experiments according to our experimental

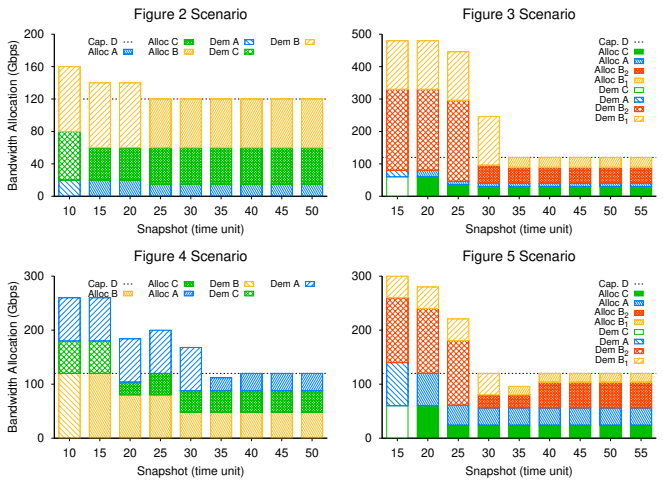

Figure 7: Measuring stability and fairness. Time units are defined as logical clock ticks in our probabilistic model.

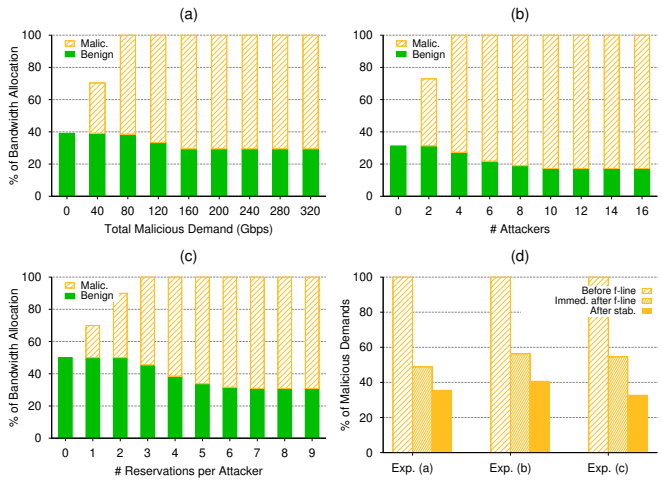

Figure 8: Measuring the impact of increasing malicious power.

goal with 100 ASes by default. Each simulation or SMC analysis took up to three hours (in the worst case) to terminate.

Experiment 1: Verifying Properties. In all our SMC analyses, the probabilities of satisfying N-Tube's properties (G1–G5) are 100%. This provides a strong independent *validation of our proofs* for the properties, and of the *model transformation* (from the LTS model into Maude), via machine-checked analysis.

Experiment 2: Stability & Fairness. We report the simulation results for the scenarios in Section III. Figure 7 depicts the bandwidth reservations at interface D and their state, demanded or allocated, as a function of (simulation) times where we take "snapshots" of the system state. The allocated bandwidths adapt over time to self-renewals and other demands. For the scenarios in Figures 3 and 5, we individually measure the allocated bandwidth for each of the two demands (B_1 and B_2) through interface B. In all scenarios, the allocations in the entire network *converge* and *stabilize*; the total allocations are always *bounded* by D's adjusted capacity (dashed line), and distributed *proportionally* to the demands after stabilization as expected (Section III). Hence, from the quantitative perspective, these results further demonstrate stability and fairness, in particular for the *worst-case scenarios* (Figures 3–5) where attacks are mounted directly on an honest path.

Experiment 3: Impact of Increasing Malicious Power. To analyze the influence on bandwidth allocation of increasing malicious power, we randomly positioned the attackers in the network (not necessarily neighbors of the targeted path), and picked a relatively small number of destinations (5 out of 100 ASes) for the reservations so that all demands, including malicious ones, *converge to these destinations*. We then randomly selected one destination and one of its egress interfaces, and reported the associated aggregate allocations for the benign sources and attackers, respectively.

Figure 8 (a–c) show the allocation percentage as a function of attacker capability, represented by *total demanded bandwidth*, *number of attackers*, and *number of issued reservations per attacker*, respectively. With increasing malicious power, the attackers tend to occupy more bandwidth until the entire allocation *stabilizes* (starting from 160 Gbps, 10 attackers, and 7 reservations per attacker, respectively); thereafter their demands are adjusted, and thus limited by the *links' capacities* and *scaling factors*. These results further provide quantitative assessments of N-Tube with varying malicious powers by exploring the large parameter space.

We also measured the adversaries' allocated bandwidth reduction by N-Tube's "frontline defense". A *frontline defender* is the first honest AS on an attacker reservation's path that can mitigate the impact of the attack by limiting the adversary demands. We divided the timeline into three phases: (i) before malicious demands reach the frontline defender; (ii) immediately after those demands break through the frontline; and (iii) after the network stabilizes.

Figure 8 (d) reports for the experiments (a–c) the percentage of original malicious demands that was allocated to the adversaries in each phase. Phase (iii)'s computation is based on the minimum stabilized "point" (e.g., 160 Gbps in experiment (a)): The higher the stabilized point is that we consider, the more reduction there will be. As demonstrated in experiment (d), N-Tube's frontline defense plays an important role in limiting the adversarial demands, e.g., in experiment (a) ∼50% of the malicious demands can be reduced, which constitute almost 80% of the total reduction.

VII. Related Work

A. Quality of Service and DDoS Protection

Congestion Control enables end-to-end connections possibly with multiple paths, to control their path rates to fairly mitigate congestion. However, this approach is based on per-flow fairness with complete knowledge of users' utility functions. In contrast, we take the stance that new Internet architectures [19], [16], [18] can handle reservation states efficiently, which allows them to police misbehaving traffic. As observed by [15], self-interested and strategic users can skew the overall rate allocation

by opening arbitrarily many connections violating the property of minimum bandwidth guarantee. Furthermore, stateless algorithms can only reduce bandwidth allocations of misbehaving flows, but cannot determine aggregated misbehavior (over time and per AS) and cannot revoke access. Note that these works do not consider an adversarial setting.

Game-Based Mechanisms consider resource allocation for maximizing a global objective function as an "inverse game theory" problem [42], [43]. Such mechanisms allow for users that are self-interested and strategic, and may attempt to manipulate the system to their advantage by misreporting information on their utility functions. The VCG type mechanisms [44] provide sealed bid auctions that incentivize users to reveal their objective truthfully and can also include sellers of resources. However, for these mechanisms to allow practical bids, the class of utility functions is very restricted, e.g., to piecewise linear [42], which misses realistic attack scenarios. Furthermore, the main objective of these mechanisms is to increase efficiency, and not to provide minimal guarantees.

Resource Reservation Systems such as RSVP [7] or RSVP-TE [45], [46], [47] enable bandwidth reservation along network paths by setting up reservation state at routers. RSVP uses soft-state reservations that require periodic updates, and must be re-established in case a path changes. However, neither do they offer authentication of reservation requests, nor handle malicious reservations, nor are their claims formally supported.

SIBRA [48] is a scalable inter-domain bandwidth allocation architecture for path-based networks. It is based on a distributed bandwidth reservation algorithm and an enforcement mechanism monitoring and policing the reservations. SIBRA is claimed to provide effective QoS guarantees in general and *minimum bandwidth guarantees* in particular. However, only a high-level design is provided without a concrete algorithm or formal arguments to support the stated claims.

B. Formal Verification of Networking Systems

As we are not aware of any work applying formal verification to a bandwidth reservation system, we discuss here research in the broader area of secure networking protocol and DoS defense verification.

Qualitative Properties. Various works study secure networking protocols, including packet forwarding protocols [49], [50], inter-domain routing protocols [51], [52], and routing protocols for mobile ad-hoc networks [53], [54], [55], [56], [57], [58], and verify their security properties such as path authorization, source authentication, path validation, route validity, and loop freedom. These works analyze *qualitative* properties using model checking, theorem proving, or hand-written proofs.

Quantitative Properties. Another critical aspect is the verification of a system's *quantitative* properties such as performance, rapid convergence, or the quick recovery from attacks. We focus on the analysis of DoS protection mechanisms. Meadows' cost-based framework [59] enables the (non-probabilistic) extension of existing protocol models and tools with cost accounting and comparisons [60]. It has been extended to cover timing aspects (e.g., slow DoS) and amplification DoS attacks [61], [33]. Approaches based on probabilistic or statistical model checking have been applied to analyze SYN flooding attacks on TCP/IP [62], the adaptive selective verification protocol [63], [64], and amplification attacks on DNS [65].

VIII. Conclusion

We have presented the design of N-Tube, along with the novel notion of bounded tube fairness. We developed formal models and verified all its safety and security properties. Moreover, we have gained: (i) additional confidence about our hand-written proofs via independent machine-checked statistical model checking, and (ii) a quantitative assessment of N-Tube's resistance to attacks by statistically exploring the large parameter space and varying malicious scenarios.

N-Tube is the first provably correct inter-domain bandwidth reservation algorithm and a major step towards a provably secure QoS scheme that also provides DDoS defense. The obvious next step is to build an efficient N-Tube implementation, as well as large-scale deployment by, e.g., proceeding along the lines proposed in [21]. Preliminary results from an N-Tube prototype implementation, realized as part of the Colibri inter-domain bandwidth-reservation infrastructure [66], have demonstrated N-Tube's deployability and scalability.

Acknowledgments

We gratefully acknowledge support for this project from the WSS Centre for Cyber Trust at ETH Zurich. We would also like to thank Stephen Shirley, Chris Pappas, Taeho Lee, Dominik Roos, Markus Legner, and Juan García-Pardo for their insightful discussions and valuable comments on how to handle tedious corner cases of the algorithm.

References

[1] M. S. Kang, S. B. Lee, and V. D. Gligor, "The Crossfire Attack," in *IEEE S&P*, 2013.

[2] A. Studer and A. Perrig, "The Coremelt attack," in *ESORICS*, 2009.

[3] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the internet: Causes, exploits, and countermeasures," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: ACM, 2014, pp. 321–333. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660299

[4] S. Shalunov and B. Teitelbaum, "Quality of service and denial of service," in *Proceedings of the ACM SIGCOMM Workshop on Revisiting IP QoS: What Have We Learned, Why Do We Care?*, ser. RIPQoS '03. New York, NY, USA: ACM, 2003, pp. 137–140. [Online]. Available: http://doi.acm.org/10.1145/944592.944600

[5] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633 (Informational), Internet Engineering Task Force, Jun. 1994. [Online]. Available: http://www.ietf.org/rfc/rfc1633.txt

[6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services," RFC 2475 (Informational), Internet Engineering Task Force, Dec. 1998, updated by RFC 3260. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[7] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network*, 1993.

[8] US-CERT, "Alert (ta17-164a) hidden cobra – north korea's ddos botnet infrastructure," https://www.us-cert.gov/ncas/alerts/TA17-164A, 2017.

[9] J. Nagle, "On Packet Switches With Infinite Storage," RFC 970, Internet Engineering Task Force, Dec. 1985. [Online]. Available: http://www.ietf.org/rfc/rfc970.txt

[10] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," *ACM SIGCOMM Comp. Comm. Rev.*, 2005.

[11] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *ACM SIGCOMM Comp. Comm. Rev.*, 1989.

[12] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks," in *ACM SIGCOMM*, 2007.

[13] J. Babiarz, K. Chan, and F. Baker, "Configuration Guidelines for DiffServ Service Classes," IETF RFC 4594.

[14] G. Hardin, "The tragedy of the commons," *Science*, 1968.

[15] B. Briscoe, "Flow rate fairness: Dismantling a religion," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, Mar. 2007. [Online]. Available: http://doi.acm.org/10.1145/1232919.1232926

[16] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: a secure internet architecture*. Springer, 2017.

[17] T. Anderson, K. Birman, R. M. Broberg, M. Caesar, D. Comer, C. Cotton, M. J. Freedman, A. Haeberlen, Z. G. Ives, A. Krishnamurthy, W. H. Lehr, B. T. Loo, D. Mazières, A. Nicolosi, J. M. Smith, I. Stoica, R. van Renesse, M. Walfish, H. Weatherspoon, and C. S. Yoo, "The NEBULA future internet architecture," in *The Future Internet - Future Internet Assembly 2013: Validated Results and New Horizons*, ser. Lecture Notes in Computer Science, A. Galis and A. Gavras, Eds., vol. 7858. Springer, 2013, pp. 16–26. [Online]. Available: https://doi.org/10.1007/978-3-642-38082-2_2

[18] P. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," in *ACM SIGCOMM Comp. Comm. Rev.*, 2009.

[19] X. Yang, D. Clark, and A. W. Berger, "Nira: A new inter-domain routing architecture," *IEEE/ACM Transactions on Networking*, 2007.

[20] J. de Ruiter and C. Schutijser, "Next-generation internet at terabit speed: SCION in P4," in *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021*, G. Carle and J. Ott, Eds. ACM, 2021, pp. 119–125. [Online]. Available: https://doi.org/10.1145/3485983.3494839

[21] C. Krähenbühl, S. Tabaeiaghdaei, C. Gloor, J. Kwon, A. Perrig, D. Hausheer, and D. Roos, "Deployment and scalability of an inter-domain multi-path routing infrastructure," in *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021*, G. Carle and J. Ott, Eds. ACM, 2021, pp. 126–140. [Online]. Available: https://doi.org/10.1145/3485983.3494862

[22] J. Naous, M. Walfish, A. Nicolosi, D. Mazières, M. Miller, and A. Seehra, "Verifying and enforcing network paths with ICING," in *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT '11, Tokyo, Japan, December 6-9, 2011*, K. Cho and M. Crovella, Eds. ACM, 2011, p. 30. [Online]. Available: http://doi.acm.org/10.1145/2079296.2079326

[23] K. Sen, M. Viswanathan, and G. Agha, "On statistical model checking of stochastic systems," in *CAV*, ser. LNCS, vol. 3576. Springer, 2005.

[24] R. Bobba, J. Grov, I. Gupta, S. Liu, J. Meseguer, P. C. Ölveczky, and S. Skeirik, "Survivability: Design, formal modeling, and validation of cloud storage systems using Maude," in *Assured Cloud Computing*. Wiley-IEEE Computer Society Press, 2018, ch. 2, pp. 10–48.

[25] S. Liu, A. Sandur, J. Meseguer, P. C. Ölveczky, and Q. Wang, "Generating correct-by-construction distributed implementations from formal Maude designs," in *NFM'20*, ser. LNCS, vol. 12229. Springer, 2020.

[26] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, *All About Maude*, ser. LNCS. Springer, 2007, vol. 4350.

[27] T. Weghorn, S. Liu, C. Sprenger, A. Perrig, and D. Basin, "N-Tube: Secure bandwidth reservation in path-aware internet architectures (supplementary material)," Available online at https://doi.org/10.5281/zenodo.5856305, 2022.

[28] S. Scherrer, C. Wu, Y. Chiang, B. Rothenberger, D. E. Asoni, A. Sateesan, J. Vliegen, N. Mentens, H. Hsiao, and A. Perrig, "Low-rate overuse flow tracer (LOFT): an efficient and scalable algorithm for detecting overuse flows," in *40th International Symposium on Reliable Distributed Systems, SRDS 2021, Chicago, IL, USA, September 20-23, 2021*. IEEE, 2021, pp. 265–276. [Online]. Available: https://doi.org/10.1109/SRDS53918.2021.00034

[29] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," RFC 6480 (Informational), Internet Engineering Task Force, Feb. 2012. [Online]. Available: http://www.ietf.org/rfc/rfc6480.txt

[30] J. Meseguer, "Conditional rewriting logic as a unified model of concurrency," *Theoretical Computer Science*, vol. 96, no. 1, pp. 73–155, 1992.

[31] A. Wang, A. J. T. Gurney, X. Han, J. Cao, B. T. Loo, C. L. Talcott, and A. Scedrov, "A reduction-based approach towards scaling up formal analysis of internet configurations," in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 2014, pp. 637–645. [Online]. Available: https://doi.org/10.1109/INFOCOM.2014.6847989

[32] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, and C. A. Gunter, "Charting the attack surface of trigger-action IoT platforms," in *CCS*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 1439–1453. [Online]. Available: https://doi.org/10.1145/3319535.3345662

[33] A. A. Urquiza, M. A. AlTurki, M. I. Kanovich, T. B. Kirigin, V. Nigam, A. Scedrov, and C. L. Talcott, "Resource-bounded intruders in denial of service attacks," in *CSF*. IEEE, 2019, pp. 382–396. [Online]. Available: https://doi.org/10.1109/CSF.2019.00033

[34] S. Liu, P. C. Ölveczky, and J. Meseguer, "Modeling and analyzing mobile ad hoc networks in Real-Time Maude," *J. Log. Algebraic Methods Program.*, vol. 85, no. 1, pp. 34–66, 2016. [Online]. Available: https://doi.org/10.1016/j.jlamp.2015.05.002

[35] S. Liu, P. C. Ölveczky, Q. Wang, I. Gupta, and J. Meseguer, "Read atomic transactions with prevention of lost updates: ROLA and its formal analysis," *Formal Asp. Comput.*, vol. 31, no. 5, pp. 503–540, 2019.

[36] S. Liu, "All in one: Design, verification, and implementation of SNOW-optimal read atomic transactions," *ACM Trans. Softw. Eng. Methodol.*, 2022. To appear.

[37] G. A. Agha, J. Meseguer, and K. Sen, "PMaude: Rewrite-based specification language for probabilistic object systems," *Electr. Notes Theor. Comput. Sci.*, vol. 153, no. 2, 2006.

[38] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *IMC'10*. ACM, 2010, pp. 267–280.

[39] CAIDA, "Topology research," https://www.caida.org/research/topology, 2021.

[40] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *OSDI*. USENIX Association, 2002.

[41] M. AlTurki and J. Meseguer, "PVeStA: A parallel statistical model checking and quantitative analysis tool," in *CALCO*, ser. LNCS, vol. 6859. Springer, 2011, pp. 386–392.

[42] R. Jain and J. Walrand, "An efficient nash-implementation mechanism for network resource allocation," *Automatica*, vol. 46, no. 8, pp. 1276–1283, 2010.

[43] R. Johari and J. N. Tsitsiklis, "Efficiency of scalar-parameterized mechanisms," *Operations Research*, vol. 57, no. 4, pp. 823–839, 2009.

[44] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *The Journal of finance*, vol. 16, no. 1, pp. 8–37, 1961.

[45] K. Shiomoto and A. Farrel, "Procedures for Dynamically Signaled Hierarchical Label Switched Paths," RFC 6107 (Proposed Standard), Internet Engineering Task Force, Feb. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6107.txt

[46] K. Kompella and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)," RFC 3477 (Proposed Standard), Internet Engineering Task Force, Jan. 2003, updated by RFC 6107. [Online]. Available: http://www.ietf.org/rfc/rfc3477.txt

[47] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions," RFC 3473 (Proposed Standard), Internet Engineering Task Force, Jan. 2003, updated by RFCs 4003, 4201, 4420, 4783, 4874, 4873, 4974, 5063, 5151, 5420, 6002, 6003, 6780. [Online]. Available: http://www.ietf.org/rfc/rfc3473.txt

[48] C. Basescu, R. M. Reischuk, P. Szalachowski, A. Perrig, Y. Zhang, H.-C. Hsiao, A. Kubota, and J. Urakawa, "Sibra: Scalable internet bandwidth reservation architecture," *arXiv preprint arXiv:1510.02696*, 2015.

[49] F. Zhang, L. Jia, C. Basescu, T. H. Kim, Y. Hu, and A. Perrig, "Mechanized network origin and path authenticity proofs," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, G. Ahn, M. Yung, and N. Li, Eds. ACM, 2014, pp. 346–357. [Online]. Available: http://doi.acm.org/10.1145/2660267.2660349

[50] T. Klenze, C. Sprenger, and D. A. Basin, "Formal verification of secure forwarding protocols," in *34th IEEE Computer Security Foundations Symposium, CSF 2021, Dubrovnik, Croatia, June 21-25, 2021*. IEEE, 2021, pp. 1–16. [Online]. Available: https://doi.org/10.1109/CSF51468.2021.00018

[51] C. Chen, L. Jia, B. T. Loo, and W. Zhou, "Reduction-based security analysis of internet routing protocols," in *20th IEEE International Conference on Network Protocols, ICNP 2012, Austin, TX, USA, October 30 - Nov. 2, 2012*. IEEE Computer Society, 2012, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/ICNP.2012.6459941

[52] C. Chen, L. Jia, H. Xu, C. Luo, W. Zhou, and B. T. Loo, "A program logic for verifying secure routing protocols," *Logical Methods in Computer Science*, vol. 11, no. 4, 2015. [Online]. Available: http://dx.doi.org/10.2168/LMCS-11(4:19)2015

[53] S. Nanz and C. Hankin, "Formal security analysis for ad-hoc networks," *Electr. Notes Theor. Comput. Sci.*, vol. 142, pp. 195–213, 2006. [Online]. Available: http://dx.doi.org/10.1016/j.entcs.2004.10.029

[54] D. Benetti, M. Merro, and L. Viganò, "Model checking ad hoc network routing protocols: ARAN vs. endaira," in *8th IEEE International Conference on Software Engineering and Formal Methods, SEFM 2010, Pisa, Italy, 13-18 September 2010*, J. L. Fiadeiro, S. Gnesi, and A. Maggiolo-Schettini, Eds. IEEE Computer Society, 2010, pp. 191–202. [Online]. Available: http://dx.doi.org/10.1109/SEFM.2010.24

[55] M. Arnaud, V. Cortier, and S. Delaune, "Deciding security for protocols with recursive tests," in *Automated Deduction - CADE-23 - 23rd International Conference on Automated Deduction, Wroclaw, Poland, July 31 - August 5, 2011. Proceedings*, ser. Lecture Notes in Computer Science, N. Bjørner and V. Sofronie-Stokkermans, Eds., vol. 6803. Springer, 2011, pp. 49–63. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-22438-6_6

[56] V. Cortier, J. Degrieck, and S. Delaune, "Analysing routing protocols: Four nodes topologies are sufficient," in *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, ser. Lecture Notes in Computer Science, P. Degano and J. D. Guttman, Eds., vol. 7215. Springer, 2012, pp. 30–50. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28641-4_3

[57] R. Chrétien and S. Delaune, "Formal analysis of privacy for routing protocols in mobile ad hoc networks," in *Principles of Security and Trust - Second International Conference, POST 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, ser. Lecture Notes in Computer Science, D. A. Basin and J. C. Mitchell, Eds., vol. 7796. Springer, 2013, pp. 1–20. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-36830-1_1

[58] M. Arnaud, V. Cortier, and S. Delaune, "Modeling and verifying ad hoc routing protocols," *Inf. Comput.*, vol. 238, pp. 30–67, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ic.2014.07.004

[59] C. A. Meadows, "A cost-based framework for analysis of denial of service networks," *Journal of Computer Security*, vol. 9, no. 1/2, pp. 143–164, 2001. [Online]. Available: http://content.iospress.com/articles/journal-of-computer-security/jcs143

[60] B. Groza and M. Minea, "Formal modelling and automatic detection of resource exhaustion attacks," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2011, Hong Kong, China, March 22-24, 2011*, B. S. N. Cheung, L. C. K. Hui, R. S. Sandhu, and D. S. Wong, Eds. ACM, 2011, pp. 326–333. [Online]. Available: https://doi.org/10.1145/1966913.1966955

[61] R. Shankesi, M. AlTurki, R. Sasse, C. A. Gunter, and J. Meseguer, "Model-checking dos amplification for voip session initiation," in *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*, ser. Lecture Notes in Computer Science, M. Backes and P. Ning, Eds., vol. 5789. Springer, 2009, pp. 390–405. [Online]. Available: https://doi.org/10.1007/978-3-642-04444-1_24

[62] G. Agha, M. Greenwald, C. A. Gunter, S. Khanna, J. Meseguer, K. Sen, and P. Thati, "Formal modeling and analysis of dos using probabilistic rewrite theories," in *International Workshop on Foundations of Computer Security (FCS '05)*, 2005.

[63] M. AlTurki, J. Meseguer, and C. A. Gunter, "Probabilistic modeling and analysis of dos protection for the ASV protocol," *Electron. Notes Theor. Comput. Sci.*, vol. 234, pp. 3–18, 2009. [Online]. Available: https://doi.org/10.1016/j.entcs.2009.02.069

[64] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A selective defense for application layer ddos attacks," in *IEEE Joint Intelligence and Security Informatics Conference, JISIC 2014, The Hague, The Netherlands, 24-26 September, 2014*. IEEE, 2014, pp. 75–82. [Online]. Available: https://doi.org/10.1109/JISIC.2014.21

[65] T. Deshpande, P. Katsaros, S. Basagiannis, and S. A. Smolka, "Formal analysis of the DNS bandwidth amplification attack and its countermeasures using probabilistic model checking," in *13th IEEE International Symposium on High-Assurance Systems Engineering, HASE 2011, Boca Raton, FL, USA, November 10-12, 2011*, T. M. Khoshgoftaar, Ed. IEEE Computer Society, 2011, pp. 360–367. [Online]. Available: https://doi.org/10.1109/HASE.2011.57

[66] G. Giuliari, D. Roos, M. Wyss, J. Á. García-Pardo, M. Legner, and A. Perrig, "Colibri: a cooperative lightweight inter-domain bandwidth-reservation infrastructure," in *CoNEXT '21: The 17th International Conference on emerging Networking EXperiments and Technologies, Virtual Event, Munich, Germany, December 7 - 10, 2021*, G. Carle and J. Ott, Eds. ACM, 2021, pp. 104–118. [Online]. Available: https://doi.org/10.1145/3485983.3494871

APPENDIX

A. Handling Node and Link Failures

In practice, source ASes can use the N-Tube algorithm to detect link failures and non-responsive nodes. In case the source AS does not receive a return message to one of its reservation requests, it can successively probe prefixes of that reservation's path by sending corresponding reservation requests with very low bandwidth demands. Depending which of these requests succeed, the source AS can identify which of the ASes on the path are not responding and by assumption **N2** in Section II-B can quickly choose an alternative path to circumvent the affected ASes.

We model link failures and non-responsive nodes using the drop event (DRP) where messages in buffers can be dropped any time, which simulates link or buffer failures. Note that properties (G1-G5) are not violated by such failures, as they are safety properties. However, we do not explicitly model node failures, where the node's reservation map becomes inconsistent. In our events, reservation maps are are persistent and updated atomically according to the N-Tube algorithm.

B. Parameters for Statistical Analysis

Table II lists all parameters used in the statistical analysis of N-Tube as described Section VI together with their default values, which the three generators use to generate the topologies, paths, and workloads (including reservations, renewals, and deletions).

Table II: Generators Parameters

Parameters	Default Value
# benign ASes	90
# malicious ASes	10
# sources	20
# intermediate ASes	75
# destinations	5
adjusted capacity δ	0.8
minBW	[0,50]
maxBW	[50,250]
maxT	20
reservation frequency	10
renewal frequency	2
deletion frequency	50
snapshot frequency	5
reservations per source	5
# paths per source-dest pair	5
segment length	5
link capacity	[100,300]
message delay	lognormal: $\mu = 0.0$, $\sigma = 1.0$