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).

NOTE. This is the full-version technical report accompanying our CSF 2022 paper.

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]

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

(which is part of the NEBULA architecture) already include packet authentication, which is needed for monitoring and 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. This report and 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 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 B.

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.

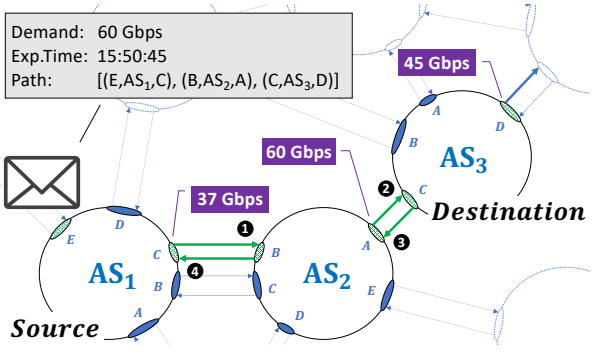


Figure 1: The process of making a reservation

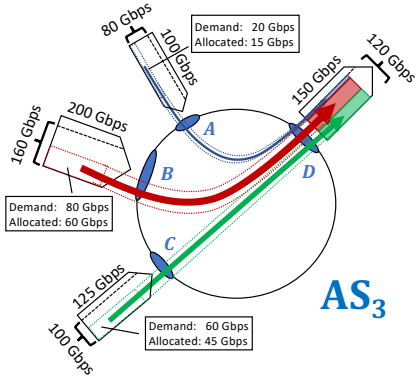


Figure 2: Bandwidth allocation computation at AS_3 distributes the egress link's (adjusted) bandwidth capacity D proportionally to three ingress demands.

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 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`) $\cdot$ 0.8 $\cdot$ `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.

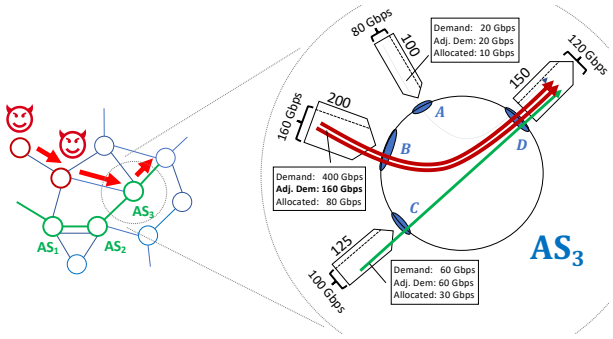


Figure 3: Limit demands on the ingress interface.

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.

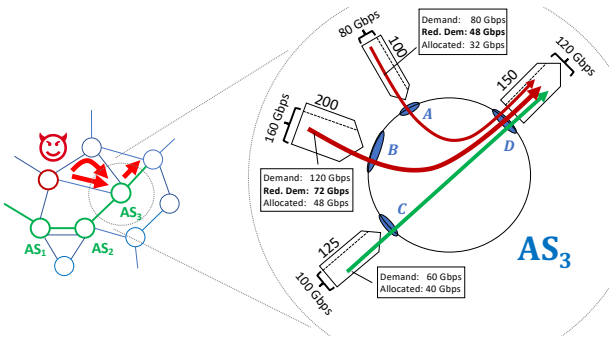


Figure 4: Limit demands on the egress interface.

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.

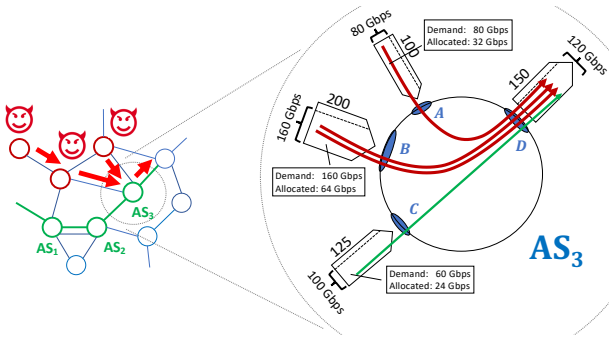


Figure 5: Worst-case minimum bandwidth guarantee.

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

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

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 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 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 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.

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

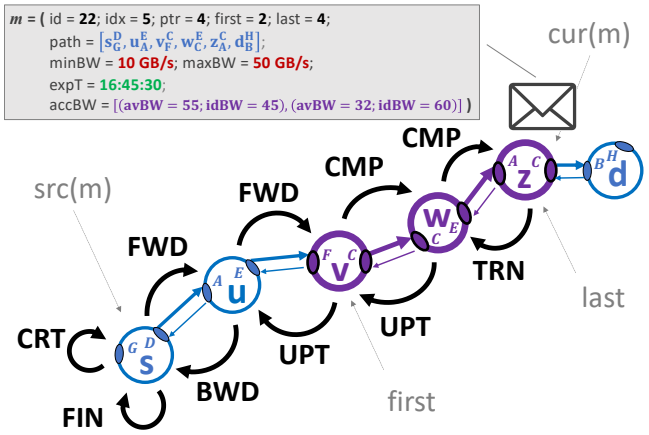


Figure 6: The message m is processed along p by the events described in Section IV-E2.

- 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* 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

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

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 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_e^i 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 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 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 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 F 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

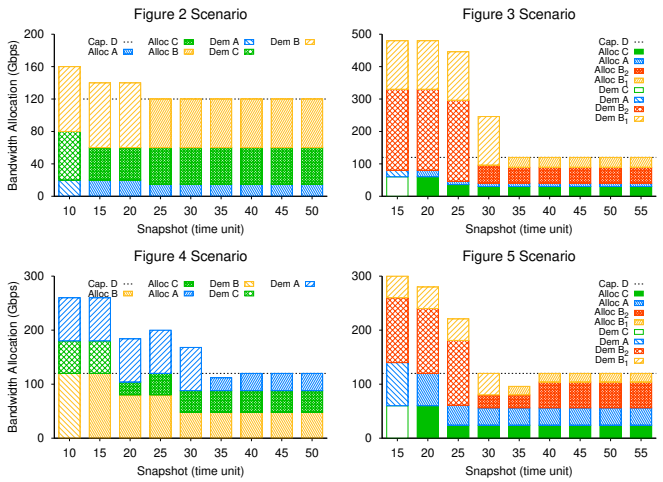


Figure 7: Measuring stability and fairness. Time units are defined as logical clock ticks in our probabilistic model.

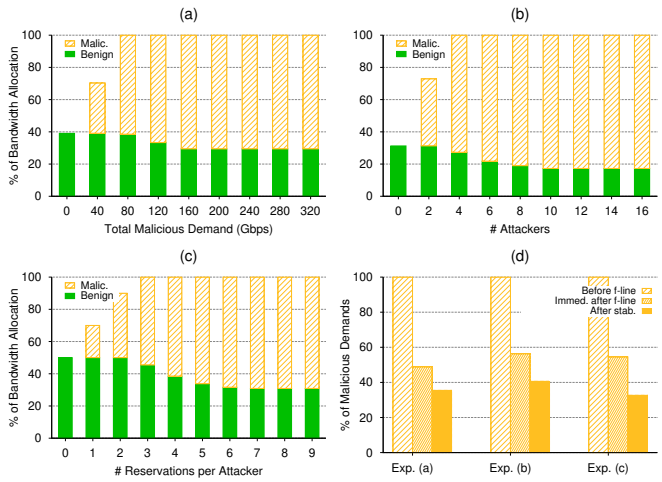


Figure 8: Measuring the impact of increasing malicious power.

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.5856306, 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. Additional Requirements

We also account for the following additional requirements:

1) N-Tube should be *efficient* in computing bandwidth allocations by using only local information of the network, i.e., based on the demands of neighbor ASes;
2) N-Tube should minimize its communication complexity by requiring only one round-trip per reservation request;
3) N-Tube should be *scalable* by reducing AS administrators' configuration efforts. In particular, bandwidth allocations should be computed automatically;
4) N-Tube should additionally allow administrators to specify bandwidth restrictions between adjacent ASes to adjust minimum bandwidth guarantees; and
5) N-Tube should provide ASes with *flexibility* by allowing them to reserve segments of paths, and to update and delete reservations.

B. 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.

C. Additional Intuition for Minimal Bandwidth Guarantee

To compute the request ratio $reqRatio_0$, we currently describe the simplest way, to take the ratio of the benign demand of `60 Gbps` and all adjusted demands *starting* from interface E. In Appendix D5 we explain in detail how we divide demands on a link into two segments:

1) The *starting segment* contains all reservations that start at this link and their adjusted demands are accumulated in *starting demands, startDem*; and
2) The *transit segment* contains all reservations that traverse this link and their adjusted demands are accumulated in *transit demands transDem*.

Each of the two segments gets allocated a fixed proportion of the link's capacity. In Appendix D5, we assume that this is exactly half the bandwidth, but this can be adapted as an additional parameter of N-Tube. By separating these two different kinds of demands into fixed bandwidth segments, we can guarantee that exceeding demands in one segment do not block out the allocations of the demands in the other segment. By providing a fixed proportion of the link's capacity for the transit segment and with the explanation in Section 3 we can provide a lower bound for the transit demands.

Note that there is still the possibility that a large number of (possibly malicious) ASes request excessive demands starting at a given link and therefore reduce the allocated bandwidth for benign demands in the starting segment. This is due to the fact that the request ratio *reqRatio* of a request is computed as the ratio of the requested demand and accumulated adjusted demands given by *startDem*. Since the demands are adjusted and therefore upper-bounded by the links' capacities, the crucial factor is the *number* of requests starting at a given link.

A simple solution to avoid this reduction of benign demands is that each AS allows only a fixed number of other, e.g., trusted, ASes to start a reservation at its links. However, this would break property (G1) for non-trusted ASes.

Alternatively, similarly as above, we can further split the starting segment into two sub-segments with a fixed portion of the segment's capacity:

(1.1) the *source segment* contains the *local requests* from the AS owning that link, i.e., the source AS of these requests, and
(1.2) the *external segment* contains the *telescoping requests* from external ASes that start their reservations at this link.

For each of these sub-segments, the request ratio of a reservation is computed as the ratio of the requested demand and the respective accumulated adjusted demands in this segment. Hence, as before between transit and starting demands, excessive demands in one of these two sub-segments cannot squeeze the allocated bandwidth in the other sub-segment.

The lower bound for the request ratio in the *source segment* is determined as the ratio of the requested demand and the starting segment's capacity multiplied by the number of reservations made by the source AS, which is exactly known to that AS. A worst case lower bound can be given, e.g., if every end-hosts in the source AS has excessive demands on the given link. Assuming n end-hosts and the starting link's capacity c_E the example in Section III, a lower bound for $reqRatio_0$ can be given by the ratio of the benign demands and the number of end hosts multiplied by the starting link's capacity, i.e., $reqRatio_0 \geq$ `60 Gbps / (`$n \cdot c_E$`)`.

Possible excessive telescoping requests from external (possibly malicious) ASes starting at the link are isolated in the *external segment*, for which no fixed lower bound guarantees can be given.

In summary, for local reservation requests by honest ASes along a path starting from one of their own links, there is a lower bound for the requested ratio. Together with the local lower bounds at each AS on the path, we can provide a global lower bound *glb* for the whole path as described in Section III.

D. Model Details

1) Network and Environment: The network is modeled as a *directed, labeled multi-graph*. We provide a more refined model using *arcs A* and two corresponding functions *src* and *tgt* to define a network:

Definition 5. Given three finite sets V, A, and I. We define a *network* $\eta \in \mathscr{N}$ as a record with

$$\mathscr{N} = (\!|\ ases = V; links = A; intf = I; \\ sft, tgt \in A \to V \times I; cap \in A \to \mathbb{R}_0^+ \ |\!)$$

with the following components:

- V is a finite set of vertices, called ASes.
- The nodes of the graph are given by the set $V \times I$.
- Hereby, the finite set I provides a global set of identifiers, which are used for interfaces inside of each AS.
- The finite set of arcs A is called *links* being the domain for the following functions:
  - The weight/label of each link is given by the function $cap : A \to \mathbb{R}_0^+$, the *capacity function*.
  - A link can start from exactly one interface of an AS given by the injective function $sft : A \to V \times I$ and
  - ends in at exactly one interface of another AS given by the injective function $tgt : A \to V \times I$.
- To guarantee that G is a valid network graph the following constraints must hold:
  - No internal links

$$\forall u, v \in V, i, j \in I, a \in A : \\ sft(a) = (u,i) \wedge tgt(a) = (v,j) \Rightarrow u \neq v$$

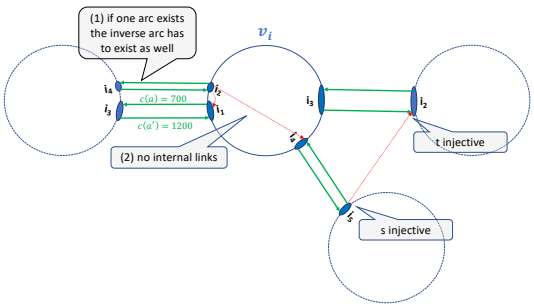
  - Inverse links

$$\forall u, v \in V, i, j \in I, a \in A. \\ sft(a) = (u,i) \wedge tgt(a) = (v,j) \\ \Rightarrow \exists a' \in A.\ sft(a') = (v,j) \wedge tgt(a') = (u,i)$$

Definition 6. The set *environment* Γ is defined as

$$\Gamma = (\!|\ \eta \in \mathscr{N}; \delta \in]0;1[; maxT, bufT \in \mathbb{R}_0^+ \ |\!).$$

An environment $\gamma \in \Gamma$ contains the network graph η, the N-Tube parameters δ, to adjust link capacities, *maxT*, limiting the maximum time a version of a reservation can be valid, and *bufT*, limiting the maximum time a message stays in a buffer until it is processed.



2) Paths: For any $p \in \mathscr{P}$ it has to hold that it

- contains at at least one link, i.e., $length(p) \geq 1$
- is directed and consistent with the network

$$\forall k \in \mathbb{N}, u, v \in V, i, e, i', e' \in I.$$
$$0 \leq k \leq length(p) - 1 \ \wedge \ p[k] = u_i^e \ \wedge \ p[k+1] = v_{i'}^{e'}$$
$$\Rightarrow \ \exists a \in A.\ src(a) = (u, e) \ \wedge \ tgt(a) = (v, i')$$

- is loop-free

$$\forall k, k' \in \mathbb{N}, u, v \in V, i, e, i', e' \in I.$$
$$k < k' \leq length(p) \ \wedge \ p[k] = u_i^e \ \wedge \ p[k'] = v_{i'}^{e'} \ \Rightarrow \ u \neq v$$

The set of ASes on a path p are defined by

$$nodes(p) := \{v \in V \mid \exists k \in \mathbb{N}.\ \ p[k].as = v\}$$

In this work we only consider the set of valid paths $\mathscr{P}$ and ignore the underlying network η.

3) Messages: As described before *messages* $\mathscr{M}$ are defined as either *deletion messages* or *reservation messages*

$$\mathscr{M} = \ \mathscr{M}_D + \mathscr{M}_R$$

together with injections $delM : \mathscr{M}_D \to \mathscr{M}$ and $resM : \mathscr{M}_R \to \mathscr{M}$.

For a valid message m the following must hold:

1) Valid path, i.e., $m.path \in \mathscr{P}$
2) Valid path counter

$$m.ptr \leq length(m.path)$$

3) Valid pointers

$$m.first < m.last \leq length(m.path)$$

4) Valid current bandwidth

$$length(m.accBW) = \max(0, m.ptr - m.first)$$

5) Valid bandwidth range

$$m.minBW \leq m.maxBW \ \wedge \ 0 < m.maxBW$$

The function $src : \mathscr{M} \to V$ extracts the source AS from a message's path.

$$src(m) = m.path[0].as$$

The function $cur : \mathscr{M} \to (\!|\ inI \in I; as \in V; egI \in I\ |\!)$ extracts the current AS together with the corresponding ingress and egress interface from a message m.

$$cur(m) = m.path[m.ptr]$$

The function $nodes : \mathscr{M} \to \mathbb{P}(V)$ extracts the AS between on the *path* field of a message,

$$nodes(m) = \{v \in V \mid \exists k \in [0; length(m.path)].\ v = m.path[k].as\}.$$

The function $sgmt : \mathscr{M}_R \to \mathbb{P}(V)$ extracts the AS between *first* and *last* of a reservation message,

$$sgmt(m) = \{v \in V \mid \exists k \in [m.first; m.last].\ v = m.path[k].as\}.$$

Given two messages $m, m' \in \mathscr{M}$ We say m *corresponds to* m' (and reversely) if they refer to the same version of a reservation

$$m \sim m' :\Leftrightarrow$$
$$src(m) = src(m') \ \wedge \ m.id = m'.id \ \wedge \ m.idx = m'.idx$$

Given two reservation messages $m, m' \in \mathscr{M}_R$. We say m *is equivalent to* m' (and reversely) if all fields except of their ptr and $accBW$ coincide

$$\begin{aligned} m \approx m' \ :\Leftrightarrow \\ & m.id = m'.id \ \wedge \\ & m.idx = m'.idx \ \wedge \\ & m.path = m'.path \ \wedge \\ & m.first = m'.first \ \wedge \\ & m.last = m'.last \ \wedge \\ & m.minBW = m'.minBW \ \wedge \\ & m.maxBW = m'.maxBW \ \wedge \\ & m.expT = m'.expT \end{aligned}$$

Note that the relations $\approx \subseteq \mathscr{M}_R \times \mathscr{M}_R$ and $\sim \subseteq \mathscr{M} \times \mathscr{M}$ are equivalence relations.

4) Reservation Maps: We model the *reservation maps* in the network as a partial function of type

$$ResMap = V \times V \times \mathbb{N} \rightharpoonup_{fin} Res.$$

A partial function $res \in ResMap$ stores for each AS x its *reservations* $res(x, s, id)$, given by the corresponding pair of reservation identifiers (s, id). Note that, compared to the previous section, we combine all reservation maps $resM_x$ into a global one, but continue using the indexed notation. Each reservation is given by a record of type

$$Res = (\!|\ path \in \mathscr{P}; ptr, first, last \in \mathbb{N}; vrs \in VrsMap\ |\!),$$

containing a *version map* of the partial function type

$$VrsMap = \mathbb{N} \rightharpoonup_{fin} (\!|\ minBW, maxBW, idBW, resBW \in \mathbb{R}_0^+;$$
$$expT \in \mathbb{N}\ |\!).$$

A valid reservation map has to be consistent regarding their reservations in the following sense:

$$\begin{aligned}
&\forall v,s \in V, id, k, st, en \in \mathbb{N}, p \in \mathscr{P}, vers \in VrsMap \\
&\quad \sigma.res(v,s,id) = (\!| \ p,k,st,en,vers \ |\!) \ \wedge \\
&\quad p[k].as = v \ \wedge \ p \in \mathscr{P} \ \wedge \ s = p[0].as \ \wedge \\
&\quad st \leq k \leq en \leq length(p) \ \wedge \\
&\quad \forall idx \in \mathbb{N}, min, max, idl, res \in \mathbb{R}_0^+, expT \in \mathbb{N}. \\
&\qquad vers(idx) = (\!| \ min, max, idl, res, expT \ |\!) \\
&\qquad min \leq res \leq max \ \wedge \ 0 < max
\end{aligned}$$

and it must hold that the reserved bandwidth for any egress link does not exceed the link's capacity, i.e.,

$$\begin{aligned}
&\forall v \in V, e \in I, t \in \mathbb{N}. \\
&\sum_{\substack{r \in rng(\sigma.res_v): \\ resEg(r)=e}} allocBW(r.vrs,t) \leq cap(v,e).
\end{aligned}$$

The functions $resSr : Res \to V$, $resEg : Res \to I$ and $resIn : Res \to I$ extract the corresponding reservation's source AS and egress and ingress interface.

$$\begin{aligned}
resSr(r) &= r.path[0].as \\
resEg(r) &= r.path[r.ptr].egI \\
resIn(r) &= r.path[r.ptr].inI
\end{aligned}$$

The function $sgmt : Res \to \mathbb{P}(V)$ ASes on the *path segment of a reservation* are given by the following function:

$$sgmt(r) = \{v \in V \mid \exists k \in [r.first; r.last]. \ v = r.path[k].as\}.$$

Given a reservation $r \in Res$ and a reservation message $m \in \mathscr{M}_R$. We say *r corresponds to m* if holds

$$\begin{aligned}
&r \propto m \ : \Leftrightarrow \\
&\quad m.maxBW = r.maxBW \ \wedge \\
&\quad m.path = r.path \ \wedge \\
&\quad m.first = r.first \ \wedge \\
&\quad m.last = r.last
\end{aligned}$$

5) Bandwidth Allocation Computation: **Available Bandwidth Computation** The function *avail* computes for message m how much bandwidth is *available* on the egress link at interface e of AS v as follows. First, $resM'$ is obtained from $resM$ by removing the reservation corresponding to m. Second, $resM'_v$ is obtained by extracting the reservations that go through AS v. Finally, *avail* subtracts the aggregated *allocated bandwidth* of all currently valid reservations with the same egress interface e from the link's total capacity $cap(x,e)$ and multiplies the result by the parameter δ to obtain the remaining bandwidth. Multiplying with $0 < \delta < 1$ guarantees that some bandwidth is always available for subsequent reservation requests.

$$\begin{aligned}
&avail(m, resM, \delta, t) = \\
&\quad \textbf{let} \\
&\quad (\!| \ i,v,e \ |\!) = cur(m) \\
&\quad resM' = resM\,((v, src(m), m.id) \mapsto \bot) \\
&\quad resM'_v = filter(resM', v) \\
&\quad \textbf{in} \\
&\quad \delta \cdot \left(cap(v,e) - \sum_{\substack{r \in rng(resM'_v): \\ resEg(r)=e}} allocBW(r.vrs,t) \right)
\end{aligned}$$

Given an AS v and a reservation $resM$ the function *filter* restricts $resM$ to the reservations that go through v.

$$\begin{aligned}
&filter(resM, v) = \\
&\quad \lambda(s', id'). \\
&\quad \textbf{let } r = resM(v, s', id') \\
&\quad \textbf{in } \left(\textbf{if } r.first \leq r.ptr \leq r.last \textbf{ then } resM(v, s', id') \textbf{ else } \bot\right)
\end{aligned}$$

Given the current time t, a *currently valid* version *vrs* is not *expired*, i.e., $vrs.expT \geq t$, and *successful*, i.e., $vrs.minBW \leq vrs.resBW$. The reservation's bandwidth *allocation* are computed by the function *allocBW* and is defined as the maximum of its currently valid versions' *resBW*.[2]

$$\begin{aligned}
&allocBW(vrsM, t) = \\
&\quad \max_{\substack{vrs \in \\ rng(vrsM)}} \{vrs.resBW \mid vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\}
\end{aligned}$$

The maximum is taken, since the source can send traffic using any existing version of its reservations. Hence, this computation guarantees that, in the worst-case, enough bandwidth is available.

Ideal Bandwidth Computation Given a message m, the function *ideal* computes how the adjusted capacity $\delta \cdot cap(v,e)$ of the egress link e of AS v is shared in a so-called *bounded tube-fair* manner among all the existing reservations at AS v with the same egress link e. First, $resM'$ is obtained from $resM$ by removing all existing versions and adding a new version corresponding to m. Removing previous versions of the reservation guarantees that the result of the *ideal* computation is not influenced by versions that are still valid and therefore simulates the ideal state where only versions of the reservation exist which correspond to m. Second, $resM'_v$ is obtained by extracting the reservations that go through AS v. Finally, *ideal* first proportionally splits the egress link's adjusted capacity between each ingress link by multiplying with *tubeRatio*, partitions the result between reservations starting and traversing the ingress link i by multiplying with *linkRatio*, and splits the result proportionally between all remaining reservations requests by multiplying with *reqRatio*.

[2]Note that $\max \emptyset = 0$.

$$
\begin{aligned}
&ideal(m, resM, \delta, t) = \\
&\ \textbf{let} \\
&\quad (\!| \ i, v, e \ |\!) = cur(m) \\
&\quad vrs' = (\!| \ minBW := m.minBW; \\
&\qquad\quad maxBW := m.maxBW; \\
&\qquad\quad idBW := \min(\delta cap(v,i), m.maxBW, preIdBW(m)); \\
&\qquad\quad resBW := m.minBW; \\
&\qquad\quad expT := m.expT \ |\!) \\
&\quad vrsM' = \emptyset\,(m.idx \mapsto vrs') \\
&\quad res' = (\!| \ path := m.path; \\
&\qquad\quad ptr := m.ptr; \\
&\qquad\quad first := m.first; \\
&\qquad\quad last := m.last; \\
&\qquad\quad vrs := vrsM' \ |\!) \\
&\quad resM' = resM\,((v, src(m), m.id) \mapsto res') \\
&\quad resM'_v = filter(resM', v) \\
&\quad tubeRatio = tubeRatio(v, i, e, resM'_v, t) \\
&\quad \textbf{if } (m.first < m.ptr) \\
&\quad \textbf{then } reqRatio = reqRatio_{transit}(v, src(m), m.id, i, resM'_v, t) \\
&\qquad\quad linkRatio = linkRatio_{transit}(v, i, resM'_v, t) \\
&\quad \textbf{else } reqRatio = reqRatio_{start}(v, src(m), m.id, i, resM'_v, t) \\
&\qquad\quad linkRatio = linkRatio_{start}(v, i, resM'_v, t) \\
&\ \textbf{in} \\
&\quad \min(\delta cap(v,i), m.maxBW, \\
&\qquad\quad reqRatio \cdot linkRatio \cdot tubeRatio \cdot \delta \cdot cap(v,e)).
\end{aligned}
$$

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 $\min\{\delta cap(v,i), tubeDem(i,e)\}$, and the aggregated bounded tube demands at e.

$$
tubeRatio(v,i,e,resM,t) = \frac{\min\{\delta cap(v,i), tubeDem(v,i,e,resM,t)\}}{\sum_{i' \in I} \min\{\delta cap(v,i'), tubeDem(v,i',e,resM,t)\}}.
$$

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

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

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

A source can demand more than the ingress and egress links' capacities allow. To account for that, the *adjusted requested demand* of a reservation r is derived from its *requested demand*, by multiplying the latter with the minimum of two *scaling factors*

$$
\begin{aligned}
&adjReqDem(v,r,resM,t) = \\
&\ \textbf{let} \\
&\quad s = resSr(r) \\
&\quad i = resIn(r) \\
&\quad e = resEg(r) \\
&\ \textbf{in} \\
&\quad \min\{inScalFctr(v,s,i,resM,t), egScalFctr(v,s,e,resM,t)\} \\
&\qquad\qquad \cdot reqDem(v,r,i,e,t).
\end{aligned}
$$

Analogously to the *allocated bandwidth allocBW* in *avail*, the *requested demand* of a reservation r is the maximum of its *demanded bandwidth demBW*

$$
\begin{aligned}
&reqDem(v,r,i,e,t) = \\
&\quad \min\{\delta cap(v,i), \delta cap(v,e), demBW(r.vrs,t)\}.
\end{aligned}
$$

Note that, the *requested demand* is bounded by the ingress and egress links' capacities. This avoids that s reserves more bandwidth in one request than physically possible, i.e., this guarantees that

$$
reqDem(v,r,i,e,t) \leq \min\{\delta cap(v,i), \delta cap(v,e)\}.
$$

Analogously to the *allocated bandwidth allocBW* in *avail*, the *requested demand* of a reservation r is the maximum of its *demanded bandwidth demBW* .

$$
\begin{aligned}
&demBW(vrsM,t) = \\
&\quad \max_{\substack{vrs \in \\ rng(vrsM)}} \{vrs.maxBW \mid vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\}
\end{aligned}
$$

The maximum is taken, since the source can send traffic using any existing version of its reservations. Hence, this computation guarantees that, in the worst-case, enough bandwidth is available.

However, it is possible that s demands less than the capacity of an ingress (or egress) link in each of its requests, but the aggregate of all its demands might still exceed the link's capacity. To adjust the *requested demands*, we multiply them with the minimum of the corresponding *ingress* and the *egress scaling factor*. We compute the *egress scaling factor* on the egress link at e for s as the source's proportion of the total *egress demand* bounded by the egress link's capacity, given by the function

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

Analogously, we define the *ingress scaling factor* on the egress link at e for s

$$inScalFctr(v,s,i,resM,t) = \frac{\min(\delta cap(v,i), inDem(v,s,i,resM,t))}{inDem(v,s,i,resM,t)}.$$

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

$$egDem(v,s,e,resM,t) = \sum_{\substack{r' \in rng(resM): \\ resSr(r')=s \\ resEg(r')=e}} reqDem(v,r',resIn(r'),e,t).$$

Analogously, we compute the source's *ingress scaling factor* on the ingress interface i,

$$inDem(v,s,i,resM,t) = \sum_{\substack{r' \in rng(resM): \\ resSr(r')=s \\ resIn(r')=i}} reqDem(v,r',i,resEg(r'),t).$$

Link Ratio: If v is a transit AS on m's path, i.e., $m.first < m.ptr$ ($\leq m.last$), then the *link ratio* between an ingress interface i and an egress interface e is computed as the ratio of the *bounded transit demand* between i and e, given by $\min\{cap(v,i), transitDem(i,e)\}$, and the sum of bounded start and bounded transit demand

$$linkRatio_{transit}(v,i,resM,t) = \textbf{let } stDem = startDem(v,i,resM,t) \textbf{ } trDem = transitDem(v,i,resM,t) \textbf{ in } \frac{\min\{\delta cap(v,i), trDem\}}{\min\{\delta cap(v,i), stDem\} + \min\{\delta cap(v,i), trDem\}}.$$

If v is the first AS on m's path, i.e., $m.first = m.ptr$, then the *link ratio* between an ingress interface i and an egress interface e is computed analogously with *startDem* in the nominator instead of *transitDem*

$$linkRatio_{start}(v,i,resM,t) = \textbf{let } stDem = startDem(v,i,resM,t) \textbf{ } trDem = transitDem(v,i,resM,t) \textbf{ in } \frac{\min\{\delta cap(v,i), stDem\}}{\min\{\delta cap(v,i), stDem\} + \min\{\delta cap(v,i), trDem\}}.$$

Taking the minimum with respect to ingress link's capacity guarantees that its respective portion for transit demand compared to demands of reservations starting at i is always bounded. This prevents that the bandwidth allocated for traversing reservations can be reduced ad infinitum by excessive reservations starting at link i and vice-versa. Here the *transit demand* at an ingress interface i is the sum of the previous *adjusted ideal bandwidth demands* of traversing reservations,

$$transitDem(v,i,resM,t) = \sum_{\substack{r \in rng(resM): \\ resIn(r)=i \\ r.first<r.ptr}} adjIdDem(v,r,resM,t).$$

Analogously, the *startDem* at an ingress interface i is the sum of the previous *adjusted ideal bandwidth demands* of starting reservations,

$$startDem(v,i,resM,t) = \sum_{\substack{r \in rng(resM): \\ resIn(r)=i \\ r.first=r.ptr}} adjIdDem(v,r,resM,t).$$

The (previous) *adjusted ideal bandwidth demand* of a reservation r is similarly defined to *adjReqDem*, the *adjusted requested demand*, with the *egress scaling factors* of the corresponding source AS and egress link

$$adjIdDem(v,r,resM,t) = \textbf{let } s = resSr(r) \textbf{ } i = resIn(r) \textbf{ } e = resEg(r) \textbf{ in } egScalFctr(v,s,e,resM,t) \cdot \min\{\delta cap(v,i), \delta cap(v,e), idBW(r.vrs,t)\}.$$

Note, that we omit the ingress scaling factor *inScalFctr*, since the *previous AS* has already applied its egress scaling factor *egScalFctr*.

The previous *ideal bandwidth allocation* of a reservation's version map is similarly defined to the *allocated bandwidth* by maximizing over the field *idBW* instead of *resBW*.

$$idBW(vrsM,t) = \max_{\substack{vrs \in \\ rng(vrsM)}} \{vrs.idBW \mid vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\}.$$

Request Ratio: If v is a transit AS on m's path, i.e., $m.first < m.ptr$ ($\leq m.last$), then the *request ratio* of a reservation identified by (s,id) at ingress interface i is the ratio between its *adjusted ideal bandwidth allocation* (provided by the predecessor on the reservation's path) and the *transit demand* at interface i

$$reqRatio_{transit}(v,s,id,i,resM,t) = \frac{adjIdDem(v,resM(v,s,id),resM,t)}{transitDem(v,i,resM,t)}.$$

Similarly, in case v is the first AS on m's path, i.e., $m.first = m.ptr$, we define the *request ratio* by

$$reqRatio_{start}(v,s,id,i,resM,t) = \frac{adjIdDem(v,resM(v,s,id),resM,t)}{startDem(v,i,resM,t)}.$$

6) State space: 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, given by its *time* field. In our model, we assume discrete time, which 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 seconds) is negligible.

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 and formally defined in Appendix D4. Finally, the field *kwl* models the *attackers' knowledge*: the set of messages created, collected, and shared by malicious ASes.

The *initial states* $\Sigma_0 = \{\sigma_0\}$ are given by the single state

$$\sigma_0 = (\!|\ time := 0; buf := \emptyset; res := \emptyset; kwl := kwl_0\ |\!),$$

where time starts at 0, the buffers and reservation map are initially empty, and the attackers' initial knowledge $kwl_0 = \{m \in \mathscr{M} \mid src(m) \in M\}$ consists of all messages with a malicious source AS.

7) Events: There are 14 events in our model. In contrast to the rest of the events, the first event *TCK* and the second event *RST* are not triggered by a message arrival, but model time progress and expiration of reservations, respectively.

In case the arrived message is a reservation message the following six events can be triggered.

The event *FWD* describes how the message is forwarded along the path until it reaches the AS where the N-Tube algorithm starts reserving bandwidth. The event *CMP* continues from there and describes how the bandwidth allocation is computed by N-Tube, how the results are added to the message and how the updated message is then forwarded until the end. The event *TRN* finishes the N-Tube computation at the last AS of the reservation and sends the updated message backwards along the path.

The event *UPT* describes how the reservations are updated in the reservation maps of each AS between the end and including the start of path on the trip back. The event *BWD* sends the message backward along the path and the event *FIN* finally drops the message at the source.

In case the arrived message is a deletion message two events are triggered. In the event *RMV* all ASes forward the deletion message along the path and delete the corresponding version of the reservation from their reservation maps. The event *DST* is triggered at the destination of the path and instead of forwarding the message it drops it.

At any arrival of an reservation or deletion messages the event *DRP* can be triggered that just drops the message without any interaction with the N-Tube algorithm. Furthermore, we define two attack events *ATK* and *CLT* as described previously.

Tick event: This is the only event that models the progress of time in the system by increasing the state's *time*-field to progress in time to its successor state.

$$\begin{aligned} TCK(t \in \mathbb{N}) = \{(\sigma, \sigma') \mid\ & \texttt{- guards -} \\ & \sigma.time = t \ \wedge \\ & \texttt{- actions -} \\ & \sigma'.time = \sigma.time + 1\ \} \end{aligned}$$

Reset event: This event models the removal of expired or unsuccessful reservations. Given by the event's parameters a reservation at an honest AS v and identified by source s, id, and idx, this event can trigger non-deterministically for the corresponding reservation. Its first guard is satisfied if the reservation has been expired before the current time of the system. Its second is satisfied is satisfied if less bandwidth was reserved than the source AS asked for.

$$\begin{aligned} RST(v \in H, s \in V, id, idx \in \mathbb{N}) = \{(\sigma, \sigma') \mid\ & \texttt{- guards -} \\ & (\ \sigma.res(v,s,id).vrs(idx).expT < \sigma.time \ \vee \\ & \ \ \sigma.res(v,s,id).vrs(idx).resBW \\ & \qquad < \sigma.res(v,s,id).vrs(idx).minBW\ ) \ \wedge \\ & \texttt{- actions -} \\ & \sigma'.res = delRes(v,s,id,idx,\sigma.res)\ \} \end{aligned}$$

The function *delRes* removes the version *idx* of the reservation identified by (s,id) from v's reservation map:

$$\begin{aligned} & delRes(v, s \in V, id, idx \in \mathbb{N}, res \in ResMap) = \\ & \quad \textbf{let} \\ & \quad\ delVrs := res(v,s,id).vrs\,(idx \mapsto \bot) \\ & \quad \textbf{in} \\ & \quad\ res\,((v,s,id) \mapsto (\!|\ vrs := delVrs\ |\!)) \end{aligned}$$

8) Event Guards: For message creation and message processing events a set of checks are executed, given in as event guards. In the following the predicates for these guards are presented:

$PathCheck : \mathscr{P} \to \mathbb{B}$ checks the validity of the path $m.path$, i.e. if the path is loop-free:

$$\begin{aligned} & PathCheck(p \in \mathscr{P}) = \\ & \forall k, k' \in \mathbb{N}, u, v \in V, i, e, i', e' \in I. \\ & \quad k < k' \leq length(p)\ \wedge\ p[k] = u_i^e\ \wedge\ p[k'] = v_{i'}^{e'}\ \Rightarrow\ u \neq v \end{aligned}$$

$Loc : \mathscr{M} \to \mathbb{B}$ checks at which part the of the path $m.path$ the AS v arrived. There are five instantiations of this predicate:

$$\begin{aligned} & atSrc(m \in \mathscr{M}) = \\ & \quad m.ptr = 0 \end{aligned}$$

$$onWay(m \in \mathscr{M}) = \\ 0 < m.ptr < m.first$$

$$atSrt(m \in \mathscr{M}) = \\ m.ptr = m.first$$

$$onPth(m \in \mathscr{M}) = \\ m.first < m.ptr < m.last$$

$$atEnd(m \in \mathscr{M}) = \\ m.ptr = m.last$$

Note that by assumption (D) all these are disjoint predicates, i.e. non of their conjunctions is satisfiable.

$Dir : V \times I \times \mathscr{M}_R \to \mathbb{B}$ checks at which border router the message m arrived compared to the provided path $m.path$ and if the length of the $accBW$ field fits the position of v on the path:

$$isFwd(v \in V, i \in I, m \in \mathscr{M}_R) = \\ m.path[m.ptr].as = v \ \wedge \\ m.path[m.ptr].inI = i \ \wedge \\ length(m.accBW) = \max(0, m.ptr - m.first)$$

$$isBwd(v \in V, i \in I, m \in \mathscr{M}_R) = \\ m.path[m.ptr].as = v \ \wedge \\ m.path[m.ptr].outI = i \ \wedge \\ length(m.accBW) = m.last - m.first$$

$$isWrg(v \in V, i \in I, m \in \mathscr{M}_R) = \\ \neg(isFwd(v,i,m) \vee isBwd(v,i,m))$$

$Rsvd : ResMap \times V \times \mathscr{M}_R \times \mathbb{N} \to \mathbb{B}$ checks if there is already a valid version of the reservation corresponding to the arrived reservation message in the reservation map. The predicate $isRsvd$ defines a valid version of a reservation r, i.e., successful and not expired.

$$isRsvd(res, v, m, t) = \\ \exists r \in Res.\ r = res(v, src(m), m.id) \ \wedge \\ \neg(0)\ r \neq \bot \ \wedge \\ \neg(5)\ r.vrs \neq \emptyset \ \wedge \\ \neg(6)\ r.vrs(m.idx) \neq \bot \ \wedge \\ (9)\ r.vrs(m.idx).expT \geq t \ \wedge \\ \neg(10)\ r.vrs(m.idx).resBW \geq r.vrs(m.idx).minBW$$

The predicate $isMrkd$ defines a marked version of a reservation r to indicate that a version of the reservation with the corresponding index $m.idx$ has been made before.

$$isMrkd(res, v, m, t) = \\ \exists r \in Res.\ r = res(v, src(m), m.id) \ \wedge \\ \neg(0)\ r \neq \bot \ \wedge \\ \neg(5)\ r.vrs \neq \emptyset \ \wedge \\ \neg(6)\ r.vrs(m.idx) \neq \bot \ \wedge \\ (9)\ r.vrs(m.idx).expT \geq t \ \wedge \\ (10')\ r.vrs(m.idx).resBW = \bot$$

The predicate $notRsvd$ indicates that no version of the reservation r exists yet or that one has existed but was unsuccessful or has been expired.

$$notRsvd(res, v, m, t) = \\ \exists r \in Res.\ r = res(v, src(m), m.id) \ \wedge \\ (\ (0)\ r = \bot \ \vee \\ (5)\ r.vrs = \emptyset \ \vee \\ (6)\ r.vrs(m.idx) = \bot \ \vee \\ \neg(9)\ r.vrs(m.idx).expT < t \ \vee \\ \neg(10)\ r.vrs(m.idx).resBW < r.vrs(m.idx).minBW\ )$$

It holds that all three event are mutual exclusive and cover all cases:

Lemma 1.

$$isMrkd \wedge isRsvd \Leftrightarrow \text{FALSE} \\ \neg(isMrkd \vee isRsvd) \Leftrightarrow notRsvd$$

$ResMsgCheck : \mathscr{M}_R \times \mathbb{N} \times \mathbb{N} \to \mathbb{B}$ is checked only if m is a reservation message as follows:

$$ResMsgCheck(m \in \mathscr{M}_R, t, maxT \in \mathbb{N}) = \\ m.expT - maxT \leq t < m.expT \ \wedge \\ m.minBW \leq m.maxBW \ \wedge \\ 0 < m.maxBW \ \wedge \\ m.first < m.last \leq length(m.path)$$

$ResMapCheck : ResMap \times V \times \mathscr{M}_R \to \mathbb{B}$ compares the $path$, and the pointers ptr, $first$ and $last$ of a message m with the corresponding reservation in res of v at the entry $m.id$ as

follows:

$$
\begin{aligned}
&\mathit{ResMapCheck}(\mathit{res}, v, m) = \\
&\quad \exists r \in \mathit{Res}.\ r = \mathit{res}(v, \mathit{src}(m), m.\mathit{id}) \ \wedge \\
&\qquad (0)\ r = \bot \ \vee \\
&\qquad (\neg(0)\ r \neq \bot \ \wedge \\
&\qquad\ (1)\ r.\mathit{path} = m.\mathit{path} \ \wedge \\
&\qquad\ (2)\ r.\mathit{ptr} = m.\mathit{ptr} \ \wedge \\
&\qquad\ (3)\ r.\mathit{first} = m.\mathit{first} \ \wedge \\
&\qquad\ (4)\ r.\mathit{last} = m.\mathit{last} \ \wedge \\
&\qquad\ ((5)\ r.\mathit{vrs} = \emptyset \ \vee \\
&\qquad\ \neg(5)\ r.\mathit{vrs} \neq \emptyset \ \wedge \\
&\qquad\ \neg(6)\ r.\mathit{vrs}(m.\mathit{idx}) \neq \bot \ \wedge \\
&\qquad\ (7)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{minBW} = m.\mathit{minBW} \ \wedge \\
&\qquad\ (8)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{minBW} = m.\mathit{maxBW} \ \wedge \\
&\qquad\ (9)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{expT} = m.\mathit{expT}\ )\ )
\end{aligned}
$$

After the message processing events on the back traversal of the path it holds that (1) the reservation message was valid and (2) there has been a corresponding reservation, i.e.,

$$\mathit{ResMapCheck}(\mathit{res}, v, m) \wedge \mathit{isRsvd}(\mathit{res}, v, m, t)$$

This can be summarized in the property:

$$
\begin{aligned}
&\mathit{isStrongRsvd}(\mathit{res}, v, m, t) = \\
&\ r = \mathit{res}(v, \mathit{src}(m), m.\mathit{id}) \ \wedge \\
&\neg(0)\ r \neq \bot \ \wedge \\
&\ (1)\ r.\mathit{path} = m.\mathit{path} \ \wedge \\
&\ (2)\ r.\mathit{ptr} = m.\mathit{ptr} \ \wedge \\
&\ (3)\ r.\mathit{first} = m.\mathit{first} \ \wedge \\
&\ (4)\ r.\mathit{last} = m.\mathit{last} \ \wedge \\
&\neg(5)\ r.\mathit{vrs} \neq \emptyset \\
&\neg(6)\ r.\mathit{vrs}(m.\mathit{idx}) \neq \bot \ \wedge \\
&\ (7)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{minBW} = m.\mathit{minBW} \ \wedge \\
&\ (8)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{maxBW} = m.\mathit{maxBW} \ \wedge \\
&\ (9)\ r.\mathit{vrs}(m.\mathit{idx}).\mathit{expT} \geq t
\end{aligned}
$$

The following Lemma shows that the predicate *isStrongRsvd* is equivalent to $\mathit{ResMapCheck}(\mathit{res}, v, m)$ together with $\mathit{isRsvd}(\mathit{res}, v, m, t)$:

Lemma 2.

$$
\begin{aligned}
&\mathit{ResMapCheck}(\mathit{res}, v, m) \wedge \mathit{isRsvd}(\mathit{res}, v, m, t) \\
&\quad \Leftrightarrow \mathit{isStrongRsvd}(\mathit{res}, v, m, t)
\end{aligned}
$$

$\mathit{ResMapCheck}_D : \mathit{ResMap} \times V \times \mathscr{M}_D \to \mathbb{B}$ is the analogous predicate for deletion messages which only keeps checks (1 – 4), since for a deletion message m does not contain the fields *minBW*, *maxBW*, and *expT*.

$$
\begin{aligned}
&\mathit{ResMapCheck}(\mathit{res}, v, m) = \\
&\quad \exists r \in \mathit{Res}.\ r = \mathit{res}(v, \mathit{src}(m), m.\mathit{id}) \ \wedge \\
&\qquad (0)\ r = \bot \ \vee \\
&\qquad (\neg(0)\ r \neq \bot \ \wedge \\
&\qquad\ (1)\ r.\mathit{path} = m.\mathit{path} \ \wedge \\
&\qquad\ (2)\ r.\mathit{ptr} = m.\mathit{ptr} \ \wedge \\
&\qquad\ (3)\ r.\mathit{first} = m.\mathit{first} \ \wedge \\
&\qquad\ (4)\ r.\mathit{last} = m.\mathit{last}\ )
\end{aligned}
$$

9) Message-Creation: The two *message creation* events describe how honest ASes create reservation and deletion messages correctly. The creation of reservation messages is defined as follows:

$$
\begin{aligned}
&\mathit{CRT}_R(\gamma \in \Gamma, m \in \mathscr{M}_R, v \in H, i \in I, t \in \mathbb{N}) \\
&= \{(\sigma, \sigma') \mid \\
&\qquad \texttt{- guards -} \\
&\qquad \sigma_{\mathit{time}} = t \ \wedge \\
&\qquad \mathit{PathCheck}(m.\mathit{path}) \ \wedge \\
&\qquad \mathit{ResMsgCheck}(m, \sigma.\mathit{time}, \gamma.\mathit{maxT}) \ \wedge \\
&\qquad \mathit{ResMapCheck}(\sigma.\mathit{res}, m, v) \ \wedge \\
&\qquad \mathit{atSrc}(\mathit{resMsg}(m)) \ \wedge \\
&\qquad \mathit{isFwd}(v, i, m) \ \wedge \\
&\qquad \neg \mathit{isMrkd}(\sigma.\mathit{res}, v, \mathit{resMsg}(m), \sigma.\mathit{time}) \ \wedge \\
&\qquad \texttt{- actions -} \\
&\qquad \sigma'.\mathit{buf} = \sigma.\mathit{buf}((v, i) \mapsto \sigma.\mathit{buf}(v, i) \cup \{m\}) \ \wedge \\
&\qquad \sigma'.\mathit{res} = \mathit{mark}(v, \sigma.\mathit{res}, m)\ \}
\end{aligned}
$$

AS v creates a *valid* reservation message m at time t and adds it to the buffer of the ingress interface i. To avoid that it creates a further reservation messages with the same identifiers $\mathit{src}(m)$, $m.\mathit{id}$, and $m.\mathit{idx}$, the source marks the corresponding version of the reservation in its reservation map when creating m. The guard $\neg \mathit{isMrkd}$ together with the function *mark* guarantee that

no duplicates are created as long as m is not expired.

$$
\begin{aligned}
&mark(v \in V, resM \in ResMap, m \in \mathscr{M}_R) := \\
&\quad \textbf{let} \\
&\quad vrs' := (\!| \; minBW := m'.minBW; \\
&\qquad\qquad maxBW := m'.maxBW; \\
&\qquad\qquad idBW := \bot; \\
&\qquad\qquad resBW := \bot; \\
&\qquad\qquad expT := m'.expT \; |\!) \\
&\quad vrsM' := resM(v, src(m), m'.id).vrs\left(m'.idx \mapsto vrs'\right) \\
&\quad res' := (\!| \; path := m'.path; \\
&\qquad\qquad ptr := m'.ptr; \\
&\qquad\qquad first := m'.first; \\
&\qquad\qquad last := m'.last; \\
&\qquad\qquad vrs := vrsM' \; |\!) \\
&\quad \textbf{in} \\
&\quad resM\left((v, src(m), m'.id) \mapsto res'\right)
\end{aligned}
$$

The creation of deletion messages is similarly defined, but less guards are sufficient:

$$
\begin{aligned}
&CRT_D(\gamma \in \Gamma, m \in \mathscr{M}_D, v \in H, i \in I, t \in \mathbb{N}) \\
&= \{(\sigma, \sigma') \mid \\
&\quad \texttt{- guards -} \\
&\quad \sigma_{time} = t \wedge \\
&\quad PathCheck(m.path) \wedge \\
&\quad atSrc(delMsg(m)) \wedge \\
&\quad isFwd(v, i, delMsg(m)) \wedge \\
&\quad \texttt{- actions -} \\
&\quad \sigma'.buf = \sigma.buf((v,i) \mapsto \sigma.buf(v,i) \cup \{m\}) \; \} \wedge \\
&\quad \sigma'.res = mark(v, \sigma.res, m) \; \}
\end{aligned}
$$

Note that we do check if there is a corresponding reservation in v's reservation map.

10) Reservation Message Arrival Events: These events are triggered by a reservation message arrival and are given by the following event template *RES*:

$$
\begin{aligned}
&RES(\gamma \in \Gamma, m, m' \in \mathscr{M}_R, v \in V, i \in I, t \in \mathbb{N}, \\
&\qquad Loc \in \mathscr{M} \to \mathbb{B}, \\
&\qquad Dir \in V \times I \times \mathscr{M}_R \to \mathbb{B}, \\
&\qquad Rsvd \in ResMap \times V \times \mathscr{M}_R \times \mathbb{N} \to \mathbb{B}, \\
&\qquad updMsg \in \mathscr{M}_R \times ResMap \times]0;1[\times \mathbb{N} \to \mathscr{M}_R, \\
&\qquad updBuf \in V \times I \times Buff \times \mathscr{M} \times \mathscr{M} \to Buff, \\
&\qquad updRes \in V \times ResMap \times \mathscr{M}_R \to ResMap) \\
&= \{(\sigma, \sigma') \mid \\
&\quad \texttt{- guards -} \\
&\quad m \in \sigma.buf(v,i) \wedge \\
&\quad \sigma.time = t \wedge \\
&\quad PathCheck(m.path) \wedge \\
&\quad ResMsgCheck(m, \sigma.time, \gamma.maxT) \wedge \\
&\quad ResMapCheck(\sigma.res, m, v) \wedge \\
&\quad Loc(resMsg(m)) \wedge \\
&\quad Dir(v, i, m) \wedge \\
&\quad Rsvd(\sigma.res, v, m, \sigma.time) \wedge \\
&\quad m' = updMsg(m, \sigma.res, \gamma.delta, \sigma.time) \wedge \\
&\quad \texttt{- actions -} \\
&\quad \sigma'.buf = updBuf(v, i, \sigma.buf, resMsg(m), resMsg(m')) \wedge \\
&\quad \sigma'.res = updRes(v, \sigma.res, m') \; \}
\end{aligned}
$$

This event-template should be understood as follows:

- A reservation message m arrives at AS v at interface i at time $\sigma.time$ to be processed, i.e., $m \in \sigma.buf(v,i)$ and $\sigma.time = t$.
- The predicates *PathCheck* and *ResMsgCheck* ensure that the path $m.path$ and the m are well-formed. The predicate *ResMapCheck* ensures that the arriving reservation message m fits an already existing reservation with ID $m.id$ in v's reservation map.
- The function *updMsg* given as a parameter creates a new reservation message m' from the arrived message m. Here the N-Tube computation is executed and the pointer $m.ptr$ is updated depending on the location and the direction of the message on the path.
- The predicates *Loc*, *Dir*, and *Rsvd* indicate at which part of the path $m.path$ the message is arrived, at which interface it arrived and if there is already an existing reservation for $m.id$ and $m.idx$ in v's reservation map.
- Using the function *updBuf* the processed message m' is either sent to the next AS on the path given by $m.path$. or is dropped after it returned to the source.
- The function *updRes* saves the reservation given by m' in v's reservation map between $m.first$ and $m.last$ on the path, and marks it otherwise.

Forward event: This event is triggered by the arrival of a reservation message m at interface i of AS v at time $\sigma.time$ at the source (but not at the start) or on the way to the start of the

reservation and there is no valid reservation in v's reservation map yet. Using the event template *RES* it is instantiated as follows:

$$\begin{aligned}&FWD(\gamma,m,m',v,i,t) =\\ &\ RES(\gamma,m,m',v,i,t,\\ &\qquad atSrc \wedge \neg atSrt) \vee onWay, isFwd, notRsvd,\\ &\qquad nothing, forward, mark)\end{aligned}$$

where the parameter *updMsg* is instantiated by the function *nothing*

$$nothing(m \in \mathscr{M}_R, res \in ResMap, \delta \in]0;1[, t \in \mathbb{N}) = m.$$

The parameter *updBuf* is instantiated by the function *forward* which models the sending of the processed message m' forward along the path by removing the received message m from v's buffer at interface i and adding m'' to buffer i'' of the next AS v''.

$$\begin{aligned}&forward(v \in V, i \in I, buf \in Buff, m \in \mathscr{M}, m' \in \mathscr{M}) =\\ &\quad \textbf{let}\\ &\quad m'' = m' \{\!|\ ptr := ptr + 1\ |\!\}\\ &\quad \{\!|\ i'', v'', e''\ |\!\} = cur(m'')\\ &\quad \textbf{in}\\ &\quad buf \begin{pmatrix}(v,i) & \mapsto & buf(v,i) \setminus \{m\}\\ (v'',i'') & \mapsto & buf(v'',i'') \cup \{m''\}\end{pmatrix}\end{aligned}$$

The parameter *updRes* is instantiated by the function *save*, which writes a new entry derived from the information given by message m in v's reservation map at entry $(src(m), m.id)$.

$$\begin{aligned}&save(v \in V, resM \in ResMap, m' \in \mathscr{M}_R) =\\ &\ \textbf{let}\\ &\ \{\!|\ i, v, e\ |\!\} = cur(m)\\ &\ finBW = \min(m'.accBW)\\ &\ vrs' = \{\!|\ minBW := m'.minBW;\\ &\qquad maxBW := m'.maxBW;\\ &\qquad idBW := \min\left(\delta cap(v,i), m'.maxBW, preIdBW(m')\right);\\ &\qquad resBW := finBW;\\ &\qquad expT := m'.expT\ |\!\}\\ &\ vrsM' = resM(v, src(m'), m'.id).vrs\left(m'.idx \mapsto vrs'\right)\\ &\ res' = \{\!|\ path := m'.path;\\ &\qquad ptr := m'.ptr;\\ &\qquad first := m'.first;\\ &\qquad last := m'.last;\\ &\qquad vrs := vrsM'\ |\!\}\\ &\ \textbf{in}\\ &\ resM\left((v, src(m'), m'.id) \mapsto res'\right).\end{aligned}$$

The function *preIdBW* extracts the ideal bandwidth computed by the previous AS from $m'.accBW$. In case there the message is at the first node, i.e., $m'.ptr = m'.first$, then the function returns $m'.maxBW$.

$$\begin{aligned}&preIdBW(m \in \mathscr{M}_R) =\\ &\quad \textbf{if } 0 \leq m.ptr - m.first - 1\\ &\quad \textbf{then } m.accBW[m.ptr - m.first - 1].idBW\\ &\quad \textbf{else } m.maxBW\end{aligned}$$

Note that *finBW* is less or equal to $m'.maxBW$ and $\delta cap(v,i)$, since $m'.accBW$ contains the ideal bandwidth computed at AS v. However, the entries in $m'.accBW$ could exceed $m'.maxBW$ and $\delta cap(v,i)$, if the previous AS is malicious.

Computation event: This event is triggered by the arrival of a reservation message m at interface i of the start AS v at time $\sigma.time$ at the start and on the path (but not at the end) before the end of the reservation and if there is no valid reservation in v's reservation map yet. Using the event template *RES* it is instantiated as follows:

$$\begin{aligned}&CMP(\gamma,m,m',v,i,t) =\\ &\ RES(\gamma,m,m',v,i,t,\\ &\qquad (atSrt \wedge \neg atEnd) \vee onPth, isFwd, notRsvd,\\ &\qquad compute, forward, save)\end{aligned}$$

where the function *compute* executes the N-Tube computations *avail* and *ideal* at v and adds them to m's field *accBW*:

$$\begin{aligned}&compute(m \in \mathscr{M}_R, res \in ResMap, \delta \in]0;1[, t \in \mathbb{N}) =\\ &\ \textbf{let}\\ &\ newBW = \{\!|\ avBW := avail(m, res, \delta, t);\\ &\qquad idBW := ideal(m, res, \delta, t)\ |\!\}\\ &\ \textbf{in}\\ &\ m\{\!|\ accBW = newBW \ \#\ m.accBW\ |\!\}.\end{aligned}$$

The parameter *updRes* is initiated by the function *save* as defined above. Note that there is no N-Tube computation necessary since m' contains the values of the *avail* and *ideal* computation due to the function *compute*.

Turn event: This event is triggered by the arrival of a message m at interface i of AS v at time $\sigma.time$ at the end (but not the source) of the reservation and if there is no valid reservation in v's reservation map yet. Using the event template *RES* it is instantiated as follows:

$$\begin{aligned}&TRN(\gamma,m,m',v,i,t) =\\ &\ RES(\gamma,m,m',v,i,t,\\ &\qquad atEnd \wedge \neg atSrc, isFwd, notRsvd,\\ &\qquad compute, backward, save)\end{aligned}$$

The parameter *updBuf* is initiated with the function *backward* which does the same as the function *forward* except sending

m in the opposite direction along the path.

$$
\begin{aligned}
&backward(v \in V, i \in I, buf \in Buff, m \in \mathscr{M}, m' \in \mathscr{M}) = \\
&\quad \textbf{let} \\
&\quad m'' = m' \langle\!| \; ptr := ptr - 1 \; |\!\rangle \\
&\quad \langle\!| \; i'', v'', e'' \; |\!\rangle = cur(m'') \\
&\quad \textbf{in} \\
&\quad buf \begin{pmatrix} (v,i) & \mapsto & buf(v,i) \setminus \{m\} \\ (v'',i'') & \mapsto & buf(v'',i'') \cup \{m''\} \end{pmatrix}
\end{aligned}
$$

Update event: This event is triggered by the arrival of a reservation message m at interface i of an AS v on the path or at the start (but not at the source) of the reservation at time $\sigma.time$, i.e. the message traverses the path backwards. Using the event template *RES* it is instantiated as follows:

$$
\begin{aligned}
&UPT(\gamma, m, m', v, i, t) = \\
&\quad RES(\gamma, m, m', v, i, t, \\
&\qquad onPth \vee (atSrt \wedge \neg atSrc), isBwd, isRsvd \vee isMrkd, \\
&\qquad nothing, backward, update)
\end{aligned}
$$

The parameter *updRes* is instantiated by the function *update*, which updates the entry derived from the information given by message m' in v's reservation map at entry $(src(m'), m'.id)$.

$$
\begin{aligned}
&update(v \in V, resM \in ResMap, m' \in \mathscr{M}_R) = \\
&\quad \textbf{let} \\
&\quad finBW = \min\left(m'.accBW\right) \\
&\quad res' = resM\left((v, src(m'), m'.id)\right) \\
&\quad vrsM' = res'.vrs \\
&\quad vrs' = vrsM'(m'.idx) \\
&\quad resBW'' = \min\left(vrs'.resBW, finBW\right) \\
&\quad vrs'' = vrs' \langle\!| \; resBW := resBW'' \; |\!\rangle \\
&\quad vrsM'' = vrsM'\left(m'.idx \mapsto vrs''\right) \\
&\quad res'' = res' \langle\!| \; vrs := vrsM'' \; |\!\rangle \\
&\quad \textbf{in} \\
&\quad resM\left((v, src(m'), m'.id) \mapsto res''\right).
\end{aligned}
$$

The field *resBW* is updated with the minimum of $m'.accBW$. Note, that $resBW'$ is limited by the previously computed *resBW* since there might be a malicious AS on the path which changed the values in *accBW* exceeding the links' capacities or *maxBW*. In contrast to *save* it is not guaranteed by the function *compute* that the last value in *accBW* is computed by the current AS.

Backward event: This event is triggered by the arrival of a reservation message m at interface i of an AS v on the way back between start and source at time $\sigma.time$. Using the event template *RES* it is instantiated[3]

$$
\begin{aligned}
&BWD(\gamma, m, m', v, i, t) = \\
&\quad RES(\gamma, m, m', v, i, t, \\
&\qquad onWay, isBwd, isRsvd \vee isMrkd, \\
&\qquad nothing, backward, update).
\end{aligned}
$$

Finish event: This event is triggered by the arrival of a reservation message m at interface i of the source (but not at the end) AS v of the path at time $\sigma.time$, i.e., the message completes its round-trip, the source updates its reservation map *res* with the final result and drops the message. Using the event template *RES* it is instantiated by

$$
\begin{aligned}
&FIN(\gamma, m, m', v, i, t) = \\
&\quad RES(\gamma, m, m', v, i, t, \\
&\qquad atSrc \wedge \neg atEnd, isBwd, isRsvd \vee isMrkd, \\
&\qquad nothing, drop, update).
\end{aligned}
$$

The parameter *updBuf* is instantiated by the function *drop*

$$
\begin{aligned}
&drop(v \in V, i \in I, buf \in Buff, m, m' \in \mathscr{M}) = \\
&\qquad buf\left((v,i) \mapsto buf(v,i) \setminus \{m\}\right).
\end{aligned}
$$

One event: This event is triggered if the reservation message m only reserves at the source AS, i.e., $m.first = m.last = 0$, at interface i at time $\sigma.time$. The message completes its round-trip at the source, which computes the bandwidth, updates its reservation map *res* with the computed result, and drops the message. Using the event template *RES* it is instantiated by

$$
\begin{aligned}
&ONE(\gamma, m, m', v, i, t) = \\
&\quad RES(\gamma, m, m', v, i, t, \\
&\qquad atSrc \wedge atEnd, isFwd, isMrkd, \\
&\qquad compute, drop, save).
\end{aligned}
$$

[3]Note, that if the reservation's field *resBW* is undefined it holds

$$resBW' = \min(vrs'.resBW, m'.accBW) = \min(m'.accBW)$$

11) Deletion Events: The two events triggered by a arrival of a deletion message are given by the following event template:

$$
\begin{aligned}
&DEL(m, m' \in \mathscr{M}_D, v \in V, i \in I, t \in \mathbb{N},\\
&\quad Loc : \mathscr{M} \to \mathbb{B},\\
&\quad updBuf : V \times I \times Buff \times \mathscr{M} \times \mathscr{M} \to Buff\\
&= \{(\sigma, \sigma') \mid\\
&\quad \texttt{- guards -}\\
&\quad m \in \sigma.buf(v,i) \wedge\\
&\quad \sigma.time = t \wedge\\
&\quad PathCheck(m.path) \wedge\\
&\quad ResMapCheck_D(m, v, \sigma.res) \wedge\\
&\quad Loc(delMsg(m)) \wedge\\
&\quad m.path[m.ptr].as = v \wedge\\
&\quad m.path[m.ptr].inI = i \wedge\\
&\quad m' = m(\!|\ ptr := ptr + 1\ |\!) \wedge\\
&\quad \texttt{- actions -}\\
&\quad \sigma'.buf = updBuf(v, i, \sigma.buf, delMsg(m), delMsg(m')) \wedge\\
&\quad \sigma'.res = remove(v, \sigma.res, m)\ \}
\end{aligned}
$$

The function *remove* overwrites an entry given by a message m in the reservation map *res* of AS v

$$
\begin{aligned}
&remove(v \in V, res \in ResMap, m \in \mathscr{M}_D) =\\
&\ \textbf{let}\\
&\ \ delVrs = res(v, src(m), m.id).vrs\,(m.idx \mapsto \bot)\\
&\ \textbf{in}\\
&\ \ res\,((v, src(m), m.id) \mapsto (\!|\ vrs := delVrs\ |\!))
\end{aligned}
$$

Note that the following parts were changed compared to the reservation message arrival template *RES*:

- Environment γ is not needed, since there is no N-Tube computation in deletion events.
- Predicate *Dir* instantiated is replaced with the first two conjuncts of predicate *isFwd*, since deletion messages only travels the path forward once and the *accBW* field is not needed in deletion messages.
- Predicate *Rsvd* is not needed, since if there is no version corresponding to m, then function *remove* does not change it.
- Function *updMsg* is replaced by the term of function *forward*.
- Function *updRes* is initiated with the function *remove*
- Predicate *ResMsgCheck* is not necessary since it validates fields of reservation messages that are not part of deletion messages.

There are two instantiations of the *DEL* event template:

Remove event: This event is triggered by the arrival of a deletion message m at interface i of AS v which is the source or on the path (but not the destination) at time $\sigma.time$. Using the event template *DEL* it is instantiated by

$$
\begin{aligned}
&RMV(m, m', v, i, t) =\\
&\ DEL(m, m', v, i, t, prePth, forward)
\end{aligned}
$$

with the path predicate $onPth : \mathscr{M} \to \mathbb{B}$

$$
\begin{aligned}
&prePth(m \in \mathscr{M}) =\\
&\ m.ptr < length(m.path).
\end{aligned}
$$

Destination event: This event is triggered by the arrival of a deletion message m at interface i of the destination AS v of the path at time $\sigma.time$. Using the event template *DEL* it is instantiated by

$$
\begin{aligned}
&DST(m, m', v, i, t) =\\
&\ DEL(m, m', v, i, t, atDst, drop).
\end{aligned}
$$

with the path predicate $atDst : \mathscr{M} \to \mathbb{B}$

$$
\begin{aligned}
&atDst(m \in \mathscr{M}) =\\
&\ m.ptr = length(m.path)
\end{aligned}
$$

12) Drop Events: This event is triggered by the arrival of a deletion or reservation message m at interface i of the source AS v of the path at time $\sigma.time$

$$
\begin{aligned}
&DRP(m \in \mathscr{M}, v \in V, i \in I, t \in \mathbb{N}) = \{(\sigma, \sigma') \mid\\
&\quad \texttt{- guards -}\\
&\quad m \in \sigma.buf(v,i) \wedge\\
&\quad \sigma.time = t \wedge\\
&\quad \texttt{- actions -}\\
&\quad \sigma'.buf = \sigma.buf(\ (v,i) \mapsto \sigma.buf(v,i) \setminus \{m\}\ )\ \}
\end{aligned}
$$

13) 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*, and (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 now discuss these two events in more detail.

Collect event: In the event *CLT*, an attacker $a \in M$ receives a message m from his buffer $buf(a,i)$ at interface i, possibly modifies its mutable fields ptr and *accBW* (but not the other fields), and adds the resulting message to *kwl*.

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

Here, the equivalence relation $m \approx m'$ expresses that m and m' coincide except on their mutable fields. This models our assumption that, in an N-Tube implementation, the source AS signs the immutable fields with its private key, while the mutable fields remain unprotected.

Attack event: In the event ATK, an attacker a can send any message m in kwl to any neighbor AS v by adding m to v's buffers $buf(v,i)$

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

These two events model powerful attack capabilities. Malicious ASes can attack anytime, make arbitrary reservation requests from their own ASes, partially modify observed requests, replay old messages, and collude by communicating 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.

Strong attack event: The attack events are given as described above. However, when Theorem 2, we assume a stronger attacker model by weakening the guards of the ATK event

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

In the previously defined ATK event we restrict by the additional guard $((a,e),(v,i)) \in E$ that v has to be honest and connected with a link to a malicious AS $a \in M$. Note, however, proving properties with this stronger attacker model does imply these properties also for the weaker one.

14) Valid Executions: **Monotonicity** Given the transition relation Δ defined in Section IV-E for any execution $\pi \in \mathfrak{E}$ is the property of *time-monotonicity*, i.e.,

Lemma 3.

$$\forall n, \tilde{n} \in \mathbb{N}. \ n \leq \tilde{n} \Rightarrow \sigma_n.time \leq \sigma_{\tilde{n}}.time$$

Proof. It's sufficient to show:

$$\forall n \in \mathbb{N}. \ \sigma_n.time \leq \sigma_{n+1}.time$$

Given $n \in \mathbb{N}$. By case distinction on λ_n:

- $TCK(\tilde{t})$: By the event's action it holds $\sigma_{n+1}.time = \sigma_n.time + 1 > \sigma_n.time$.
- All other event's actions keep the time unchanged, i.e., $\sigma_n.time = \sigma_{n+1}.time$.

□

Time-progress The global time infinitely progresses, i.e.,

$$\forall t \in \mathbb{N} \ \exists n \in \mathbb{N}. \ \sigma_n.time \geq t, \tag{TP}$$

more well-known as the property of *zeno-freeness*.

This is reasonable to assume, since it is equivalent to the assumption that in a realistic execution there are only finitely many ATK and CRT events at any given point in time

$$\begin{aligned}&\forall t \in \mathbb{N} \ \exists B \in \mathbb{N}. \\ &\quad |\{n \in \mathbb{N} \mid \lambda_n.time = t \wedge \lambda_n.event = ATK\}| \leq B \ \wedge \\ &\quad |\{n \in \mathbb{N} \mid \lambda_n.time = t \wedge \lambda_n.event = CRT\}| \leq B.\end{aligned}$$

Message-Progress: All messages in the buffers of honest ASes are processed in at most time $bufT$

$$\begin{aligned}&\forall n \in \mathbb{N}, v \in H, i \in I, m \in \sigma_n.buf(v,i). \\ &\quad \exists \tilde{n} > n. \ m \notin \sigma_{\tilde{n}}.buf(v,i) \ \wedge \ \sigma_{\tilde{n}}.time - \sigma_n.time \leq bufT.\end{aligned} \tag{MP}$$

Distinct-Pointers: W.l.o.g., we assume that all messages m their field $first$ is positive which is given as Assumption DP

$$\begin{aligned}&\forall n \in \mathbb{N}, v \in V, i \in I, m \in \mathscr{M}. \\ &\quad m \in \sigma_n.buf(v,i) \Rightarrow 0 < m.first\end{aligned} \tag{DP}$$

Note that we assumed $m.first < m.last$. Adding this assumption does not change any of the properties we prove but avoids considering additional corner-cases.

Definition (Valid Executions)**.** An execution $\pi \in \mathfrak{E}$ is *valid* if it satisfies time progress (TP), message progress (MP), and (DP) properties.

This is satisfied if (i) all honest ASes run a fair scheduling algorithm (e.g., Round-Robin) to prevent message starvation, and (ii) messages are dropped in case of buffer overflow.

We will show that the global properties (G1–G5) of the N-Tube algorithm hold for all valid executions.

E. Properties and Proofs

In this section, we define and prove the N-Tube's properties.

1) Successful Reservations and Constant Demands: Properties (G1) and (G2) assume that a *successful reservation* has been established by an honest source AS and properties (G3–G5) assume *constant demands*. We define these two notions in the following.

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* if the following three conditions hold:

- **Honest Path:** m's path only contains honest ASes.

$$nodes(m) \subseteq H. \tag{HP}$$

- **Confirmation:** s confirms m at time t with sufficient bandwidth

$$\exists n \in \mathbb{N}, i \in I. \ \lambda_n = FIN(m,s,i,t) \wedge finBW(m) \geq m.minBW. \tag{CF}$$

- **No deletion:** There is no deletion event matching the reservation $(src(m), m.id)$ and version $m.idx$ before m expires

$$\begin{aligned}&\forall \hat{n}, n_c \in \mathbb{N}, \hat{v} \in V, \hat{i}, i \in I, \hat{m} \in \mathscr{M}_D, m_c \in \mathscr{M}_R, \hat{t}, t_c \in \mathbb{N}.\\ &\ \lambda_{n_c} = CRT_R(m_c, s, i, t_c) \wedge m_c \approx m \ \wedge\\ &\ \sigma_{\hat{n}}.time \in \,]t_c; m.expT] \ \wedge\\ &\ \big(\lambda_{\hat{n}} = RMV(\hat{v}, \hat{i}, \hat{m}, \hat{t}) \ \vee \ \lambda_{\hat{n}} = DST(\hat{v}, \hat{i}, \hat{m}, \hat{t})\big)\\ &\quad \Rightarrow m_c \not\sim \hat{m}. \end{aligned} \tag{nDE}$$

Constant Demands We model "constant bandwidth demands" using a function

$$D : V \times \mathbb{N} \rightharpoonup_{fin} \mathscr{M}_R$$

such that $D(s,id) = m$ implies $src(m) = s$ and $m.id = id$, and $D(s,id).minBW = 0$.

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*, and *accBW*.

$$\begin{aligned}&m \vdash D \ : \Leftrightarrow\\ &\quad D(src(m), m.id).path = m.path \ \wedge\\ &\quad D(src(m), m.id).first = m.first \ \wedge\\ &\quad D(src(m), m.id).last = m.last \ \wedge\\ &\quad D(src(m), m.id).minBW = m.minBW \ \wedge\\ &\quad D(src(m), m.id).maxBW = m.maxBW\end{aligned}$$

We say that a *reservation r identified by (s,id) corresponds to D at time t* if $(s,id) \in supp(D)$ and r coincides with $D(s,id)$ on all fields except *ptr*, *expT*, for all its versions and *accBW*.

$$\begin{aligned}&r, t \vdash D \ : \Leftrightarrow\\ &\quad (s,id) \in supp(D) \ \wedge\\ &\quad D(s,id).path = r.path \ \wedge\\ &\quad D(s,id).first = r.first \ \wedge\\ &\quad D(s,id).last = r.last \ \wedge\\ &\quad \forall idx \in \mathbb{N}.\\ &\qquad r.vrs(idx) = \bot \ \vee\\ &\qquad r.vrs(idx).expT < t \ \vee\\ &\qquad D(s,id).minBW = r.vrs(idx).minBW \ \wedge\\ &\qquad D(s,id).maxBW = r.vrs(idx).maxBW \ )\end{aligned}$$

We say that a *state σ corresponds to D*, written $\sigma \vdash D$, if, for all honest ASes $v \in H$, the all reservations in res_v correspond to D at time $\sigma.time$, i.e.,

$$\forall v \in H, r \in rng(\sigma.res_v)\,.\, r, \sigma.time \vdash D$$

For the rest of this section we fix two time points $t_0, t_1 \in \mathbb{N}$ such that $t_1 - t_0 \geq 2maxT$. We say an execution $\pi \in \mathfrak{E}$ *has constant demands D between t_0 and t_1* if

- **Successful Requests:** 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

$$\begin{aligned}&\forall m \in rng(D), evt \in \{CMP, TRN, UPT\}, v \in sgmt(m).\\ &\ \exists \tilde{n} \in \mathbb{N}, \tilde{m}, \tilde{m}' \in \mathscr{M}_R, \tilde{i} \in I, \tilde{t} \in \mathbb{N}.\\ &\ \sigma_{\tilde{n}}.time \leq t_0 \ \wedge \ \tilde{m} \vdash D \ \wedge\\ &\ \lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{m}', v, \tilde{i}, \tilde{t}) \ \wedge \ \tilde{m}.expT > t_0,\end{aligned}$$

- **Successful Renewal:** and successfully renews this reservation without any gaps until t_1,

$$\begin{aligned}&\forall \tilde{n} \in \mathbb{N}, evt \in \{CMP, TRN, UPT\}.\ \sigma_{\tilde{n}}.time \in [t_0, t_1] \Rightarrow\\ &\ \forall \tilde{m}, \tilde{m}' \in \mathscr{M}_R, \tilde{v} \in H, \tilde{i} \in I, \tilde{t} \in \mathbb{N}.\\ &\ \lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t}) \ \wedge \ \tilde{m} \vdash D \ \Rightarrow\\ &\ \exists \hat{n} \in \mathbb{N}, \hat{m}, \hat{m}' \in \mathscr{M}_R, \tilde{v} \in sgmt(\hat{m}), \hat{i} \in I, \hat{t} \in \mathbb{N}.\\ &\ \sigma_{\hat{n}}.time \in [\tilde{t}, \tilde{m}.expT] \ \wedge\\ &\ \lambda_{\hat{n}} = evt(\hat{m}, \hat{m}', \hat{v}, \hat{i}, \hat{t}) \wedge \ \hat{m}.expT > \tilde{m}.expT.\end{aligned}$$

- **Constant Demands:** any reservation confirmed by a reservation message m between t_0 and t_1, corresponds to D

$$\begin{aligned}&\forall \tilde{n} \in \mathbb{N}, evt \in \{CMP, TRN, UPT\}.\ \sigma_{\tilde{n}}.time \in [t_0, t_1] \Rightarrow\\ &\ \forall \tilde{m}, \tilde{m}' \in \mathscr{M}_R, \tilde{i} \in I, \tilde{t} \in \mathbb{N}, \tilde{v} \in sgmt(\tilde{m}).\\ &\ \lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t}) \Rightarrow \tilde{m} \vdash D\end{aligned}$$

and similarly for attack events

$$\begin{aligned}&\forall \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time \in [t_0, t_1] \Rightarrow\\ &\ \forall \tilde{m} \in \mathscr{M}_R, \tilde{i}, \tilde{e} \in I, \tilde{t} \in \mathbb{N}, \tilde{a} \in M, \tilde{v} \in sgmt(\tilde{m}).\\ &\ \lambda_{\tilde{n}} = ATK(\tilde{m}, \tilde{a}, \tilde{v}, \tilde{i}, \tilde{e}, \tilde{t})\\ &\ \Rightarrow \tilde{m} \vdash D \ \wedge \ finBW(\tilde{m}) = D(src(\tilde{m}), \tilde{m}.id).maxBW\end{aligned}$$

- **No Deletion:** there are no deletion events between t_0 and t_1 for reservations given by $supp(D)$.

$$\begin{aligned}&\forall \tilde{n} \in \mathbb{N}, evt \in \{RMV, DST\}.\ \sigma_{\tilde{n}}.time \in [t_0, t_1]\\ &\ \forall \tilde{m} \in \mathscr{M}_D, \tilde{v} \in H, \tilde{i} \in I, \tilde{t} \in \mathbb{N}.\\ &\ \lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{v}, \tilde{i}, \tilde{t}) \Rightarrow (src(\tilde{m}), \tilde{m}.id) \notin supp(D)\end{aligned}$$

2) Successful Reservation Theorem:

Theorem 2 (Successful Reservation)**.** If an AS s makes a successful reservation m and time t, then all ASes on m's path added their *avail* and *ideal* computations to $m.accBW$ and reserve $finBW(m)$ until it expires

$$\begin{aligned}&\forall n \in \mathbb{N}, v \in V, k \in [m.first; m.last].\\ &\ \sigma_n.time \in \,]t; m.expT] \ \wedge \ v = m.path[k].as\\ &\ \Rightarrow \sigma_n.res(v, s, m.id).vrs(m.idx).resBW = finBW(m) \ \wedge\\ &\quad \exists \tilde{n} < n, \tilde{m} \in \mathscr{M}_R.\ \tilde{m} \approx m \ \wedge\\ &\qquad m.accBW[k - m.first] = (\!|\ avBW = avail(\tilde{m}, \sigma_{\tilde{n}}.res);\\ &\qquad\qquad idBW = ideal(\tilde{m}, \sigma_{\tilde{n}}.res)\ |\!).\end{aligned}$$

Proof. First we prove that

$$\begin{aligned}&\forall n \in \mathbb{N}, v \in V, k \in [m.first; m.last].\\ &\quad \sigma_n.time \in \,]t; m.expT] \;\wedge\; v = m.path[k].as\\ &\quad \Rightarrow \sigma_n.res(v,s,m.id).vrs(m.idx).resBW = finBW(m).\end{aligned}$$

Given Assumption CF, we show that the *FIN* event is preceded by an *BWD* event on the previous AS. Given a message $m \in buf(v,i)$ with $v \in nodes(m)$, we can show by induction that there was corresponding message processing event in the time interval $[m.expT - maxT, t]$ with a corresponding message $\tilde{m}$

$$\begin{aligned}&\forall v \in nodes(m).\; \exists evt \in \{CRT_R, FWD, CMP, TRN, UPT, BWD\}.\\ &\exists \tilde{n} \in \mathbb{N}, \tilde{m}, \tilde{m}' \in \mathscr{M}_R, \tilde{i} \in I, \tilde{t} \in \mathbb{N}.\\ &\sigma_{\tilde{n}}.time < t \wedge \lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{m}', v, \tilde{i}, \tilde{t}) \wedge \tilde{m} \approx m\end{aligned}$$

In the case distinction we need to exclude that the attacker put m into the buffer by using Lemma 8 which is based on our Assumption HP.

Next, we show that each of these events is unique for each successful reservation with message m

$$\begin{aligned}&\forall \tilde{n}, \hat{n} \in \mathbb{N}, evt \in \{CRT_R, FWD, CMP, TRN, UPT, BWD\},\\ &\forall \tilde{m}, \tilde{m}', \hat{m}, \hat{m}' \in \mathscr{M}_R, v \in H, \tilde{i}, \hat{i} \in I, \tilde{t}, \hat{t} \in \mathbb{N}.\\ &\lambda_{\tilde{n}} = evt(\tilde{m}, \tilde{m}', v, \tilde{i}, \tilde{t}) \wedge \lambda_{\hat{n}} = evt(\hat{m}, \hat{m}', v, \hat{i}, \hat{t}) \wedge\; \tilde{m} \approx \hat{m}\\ &\quad \Rightarrow \tilde{n} = \hat{n}\end{aligned}$$

and therefore exclude that there was another corresponding message processing event that changes the reservation done by $\tilde{m}$.

Using Assumption nDE, we can also exclude that the reservation gets deleted in the time interval $[m.expT - maxT, t]$.

Thus, for each successful reservation with a message m, we obtain a list of transitions $l_m = [\lambda_1, \ldots, \lambda_{2 \cdot m.last}]$ ordered by time, where each transition corresponds to the message unique processing event for m. Given l_m we can finally show by induction on its length

$$\begin{aligned}&\exists \tilde{n} < n, \tilde{m} \in \mathscr{M}_R.\; \tilde{m} \approx m \;\wedge\\ &\quad m.accBW[k - m.first] = (\!|\; avBW = avail(\tilde{m}, \sigma_{\tilde{n}}.res);\\ &\qquad\qquad idBW = ideal(\tilde{m}, \sigma_{\tilde{n}}.res) \;|\!).\end{aligned}$$

□

Consistency The following technical lemma shows that after a CRT_R event with message m the corresponding version is marked or reserved and remains unchanged until it expires.

Lemma 4 (CRT-isMrkd-until-expiration)**.**

$$\begin{aligned}&\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in V, i \in I, t \in \mathbb{N}.\\ &\quad \lambda_n = CRT(m,v,i,t) \;\Rightarrow\\ &\quad\; \forall \hat{n} > n.\; \sigma_{\hat{n}}.time \le m.expT \Rightarrow\\ &\qquad (\; isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\\ &\qquad\quad \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\;) \wedge\\ &\qquad ResMapCheck(\sigma_{\hat{n}}.res, v, m)\end{aligned}$$

Proof. Given $n \in \mathbb{N}$, $m, m' \in \mathscr{M}_R$ with m, $v \in V$, $i \in I$, $t \in \mathbb{N}$ with

$$\lambda_n = CRT(m,v,i,t)$$

By induction on $\hat{n} > n$:

- $\hat{n} = n+1$: Using assumption $\lambda_n = CRT(m,v,i,t)$ we can show that

  $$isMrkd(\sigma_{n+1}.res, v, m, \sigma_{n+1}.time)$$

  and

  $$ResMapCheck(\sigma_{n+1}.res, v, m).$$

  Hence, in this case ($\hat{n} = n+1$) the claim holds.
- $\hat{n} \to \hat{n}+1$: By *IH* it holds

  $$\begin{aligned}&\hat{n} > n \;\wedge\; \sigma_{\hat{n}}.time \le m.expT \Rightarrow\\ &(\; isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\\ &\quad \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\;) \wedge\\ &ResMapCheck(\sigma_{\hat{n}}.res, v, m)\end{aligned}$$

  We need to show:

  $$\begin{aligned}&\hat{n}+1 > n \;\wedge\; \sigma_{\hat{n}+1}.time \le m.expT \Rightarrow\\ &(\; isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time)\\ &\quad \vee isRsvd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time)\;) \wedge\\ &ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)\end{aligned}$$

  Assume $\hat{n}+1 > n+1$ and $\sigma_{\hat{n}+1}.time \le m.expT$.
  Case distinction by $\lambda_{\hat{n}}$:
  - $TCK(\tilde{t})$: By the event's guard it holds $\sigma_{\hat{n}}.time = \tilde{t}$. Two cases:
    * $\sigma_{\hat{n}}.time \ge m.expT$: In this case

      $$\sigma_{\hat{n}+1}.time = \sigma_{\hat{n}}.time + 1 > m.expT$$

      hence, the premise is not satisfied, i.e. the claim is true.
    * $\sigma_{\hat{n}}.time < m.expT$:
      In this case it immediately holds $\sigma_{\hat{n}}.time \le m.expT$ and we can apply *IH* and get:

      $$\begin{aligned}&(\; isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\\ &\quad \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\;) \wedge\\ &ResMapCheck(\sigma_{\hat{n}}.res, v, m)\end{aligned}$$

      Since the event's action does not change the reservation maps, i.e. $\sigma_{\hat{n}}.res = \sigma_{\hat{n}+1}.res$, it follows that

      $$\begin{aligned}&(\; isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}}.time)\\ &\quad \vee isRsvd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}}.time)\;) \wedge\\ &ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)\end{aligned}$$

      By (9) of $ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)$ and assumption $\sigma_{\hat{n}+1}.time \le m.expT$ it follows that

      $$\begin{aligned}(9')\;\; &\sigma_{\hat{n}+1}(v, src(m), m.id).vrs(m.idx).expT\\ &=^{(9)} m.expT \ge \sigma_{\hat{n}+1}.time\end{aligned}$$

i.e. $isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}}.time)$ and therefore the claim.

– $RST(\tilde{v}, \tilde{s}, \tilde{id}, \tilde{idx})$: The event's actions do not change time, i.e. $\sigma_{\hat{n}+1}.time = \sigma_{\hat{n}}.time$. Together with the assumptions $\hat{n}+1 > n+1$ and $\sigma_{\hat{n}+1}.time \leq m.expT$ we get

$$\hat{n} > n$$
$$\sigma_{\hat{n}}.time \leq m.expT$$

and can apply *IH* and get:

$$(-) \quad (\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \\ \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\ ) \wedge \\ ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$

and applying $\sigma_{\hat{n}+1}.time = \sigma_{\hat{n}}.time$ again gives us:

$$(+) \quad (\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time) \\ \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time)\ ) \wedge \\ ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$

Regarding the reservation map there are two cases:

∗ $v = \tilde{v} \wedge src(m) = \tilde{s} \wedge m.id = \tilde{id} \wedge i\tilde{d}x = m.idx$: The event's guard only deletes the corresponding version of the reservation if holds that

$$\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx).expT < \sigma_{\hat{n}}.time$$

By (9) in $(+)$ $ResMapCheck(\sigma_{\hat{n}}.res, v, m)$ it holds:

$$(9)\ res(v, src(m), m.id).vrs(m.idx).expT = m.expT$$

and by assumption $\sigma_{\hat{n}+1}.time \leq m.expT$ it holds

$$m.expT \geq \sigma_{\hat{n}+1}.time = \sigma_{\hat{n}}.time$$

i.e. a contradiction.

∗ *Otherwise* : The event's actions do not affect the version of the reservation corresponding m, i.e.

$$\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\ = \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)$$

and therefore the claim follows by $(+)$.

– $CRT_R(\tilde{m}, \tilde{v}, \tilde{i}, \tilde{t})$: As in RST it follows $(+)$, hence in particular

$$(\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \\ \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\ ) \wedge \\ ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$

The following cases:

∗ $\tilde{m} \approx m \wedge \tilde{m}.ptr = m.ptr \wedge \tilde{v} = v$: We can show that

$$isMrkd(\sigma_{\hat{n}+1}.res, v, \tilde{m}, \sigma_{\hat{n}+1}.time) \\ ResMapCheck(\sigma_{\hat{n}+1}.res, v, \tilde{m})$$

Together with $\tilde{m} \approx m$ and $\tilde{m}.ptr = m.ptr$ it follows that

$$isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time) \\ ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)$$

∗ $(\tilde{m} \not\approx m \vee \tilde{m}.ptr \neq m.ptr) \wedge \tilde{v} = v$: $\tilde{m} \not\approx m \vee \tilde{m}.ptr \neq m.ptr$ implies

$$(a)\ src(m) \neq src(\tilde{m}) \vee \\ (b)\ m.id \neq \tilde{m}.id \vee \\ (c)\ m.idx \neq \tilde{m}.idx \vee \\ (d)\ m.path \neq \tilde{m}.path \vee \\ (e)\ m.first \neq \tilde{m}.first \vee \\ (e)\ m.last \neq \tilde{m}.last \vee \\ (g)\ m.minBW \neq \tilde{m}.minBW \vee \\ (h)\ m.maxBW \neq \tilde{m}.maxBW \vee \\ (i)\ m.expT \neq \tilde{m}.expT \vee \\ (j)\ m.ptr \neq \tilde{m}.ptr$$

Two cases:

· $(a) \vee (b) \vee (c)$: The event's action do not affect the version of the reservation corresponding to m, i.e.

$$\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\ = \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)$$

and the claim follows with $(+)$.

· *Otherwise* : I.e.

$$src(m) = src(\tilde{m}) \wedge \\ m.id = \tilde{m}.id \wedge \\ m.idx = \tilde{m}.idx$$

but at least one of the other cases α in $(d), \ldots, (j)$ does not hold.
By $(+)$ holds

$$(\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \\ \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time)\ ) \wedge \\ ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$

hence

$$r = \sigma_{\hat{n}}.res(v, src(m), m.id) \wedge \\ \neg(0)\ r \neq \bot \wedge \\ (1)\ r.path = m.path \wedge \\ (2)\ r.ptr = m.ptr \wedge \\ (3)\ r.first = m.first \wedge \\ (4)\ r.last = m.last \wedge \\ \neg(5)\ r.vrs \neq \emptyset \\ \neg(6)\ r.vrs(m.idx) \neq \bot \wedge \\ (7)\ r.vrs(m.idx).minBW = m.minBW \wedge \\ (8)\ r.vrs(m.idx).maxBW = m.maxBW \wedge \\ (9)\ r.vrs(m.idx).expT = m.expT$$

But this is in contradiction to the event's guard

$$ResMapCheck(\sigma_{\hat{n}}.res, v, \tilde{m})$$

(using $\tilde{v} = v$) and the case α that does not hold, i.e.

$$\begin{aligned}
&\sigma_{\hat{n}}.res(v, src(m), m.id) \\
&\quad = \sigma_{\hat{n}}.res(v, src(\tilde{m}), \tilde{m}.id) \wedge \\
&\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\
&\quad = \sigma_{\hat{n}}.res(v, src(\tilde{m}), \tilde{m}.id).vrs(\tilde{m}.idx) \wedge \\
&\sigma_{\hat{n}}.res(v, src(m), m.id).\alpha = m.\alpha \wedge \\
&\sigma_{\hat{n}}.res(v, src(\tilde{m}), \tilde{m}.id).\alpha = \tilde{m}.\alpha \wedge \\
&m.\alpha \neq \tilde{m}.\alpha
\end{aligned}$$

Hence this guard is not satisfied and the event could not happen.

* $\tilde{v} \neq v$: The event's actions do not affect version of the reservation corresponding m, i.e.

$$\begin{aligned}
&\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\
&\quad = \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)
\end{aligned}$$

and therefore the claim follows by $(+)$.

– $CRT_D(\tilde{m}, \tilde{v}, \tilde{i}, \tilde{t})$: The event does not affect the global time and the reservation map, hence the claim follows by *IH*.

– $FWD(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t})$: As in RST it follows $(+)$, hence in particular

$$\begin{aligned}
&(\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \\
&\quad \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \ ) \wedge \\
&ResMapCheck(\sigma_{\hat{n}}.res, v, m)
\end{aligned}$$

The following cases:

* $\tilde{m} \approx m \wedge \tilde{m}.ptr = m.ptr \wedge \tilde{v} = v$: We can show that

$$\begin{aligned}
&isMrkd(\sigma_{\hat{n}+1}.res, v, \tilde{m}, \sigma_{\hat{n}+1}.time) \\
&ResMapCheck(\sigma_{\hat{n}+1}.res, v, \tilde{m})
\end{aligned}$$

Together with $\tilde{m} \approx m$ and $\tilde{m}.ptr = m.ptr$ it follows that

$$\begin{aligned}
&isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time) \\
&ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)
\end{aligned}$$

* $(\tilde{m} \not\approx m \vee \tilde{m}.ptr \neq m.ptr) \wedge \tilde{v} = v$: $\tilde{m} \not\approx m \vee \tilde{m}.ptr \neq m.ptr$ implies

$$\begin{aligned}
&(a) \ src(m) \neq src(\tilde{m}) \ \vee \\
&(b) \ m.id \neq \tilde{m}.id \ \vee \\
&(c) \ m.idx \neq \tilde{m}.idx \ \vee \\
&(d) \ m.path \neq \tilde{m}.path \ \vee \\
&(e) \ m.first \neq \tilde{m}.first \ \vee \\
&(e) \ m.last \neq \tilde{m}.last \ \vee \\
&(g) \ m.minBW \neq \tilde{m}.minBW \ \vee \\
&(h) \ m.maxBW \neq \tilde{m}.maxBW \ \vee \\
&(i) \ m.expT \neq \tilde{m}.expT \ \vee \\
&(j) \ m.ptr \neq \tilde{m}.ptr
\end{aligned}$$

Two cases:

· $(a) \vee (b) \vee (c)$: The event's action do not affect the version of the reservation corresponding to m, i.e.

$$\begin{aligned}
&\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\
&\quad = \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)
\end{aligned}$$

and the claim follows with $(+)$.

· *Otherwise* : I.e.

$$\begin{aligned}
&src(m) = src(\tilde{m}) \wedge \\
&m.id = \tilde{m}.id \wedge \\
&m.idx = \tilde{m}.idx
\end{aligned}$$

but at least one of the other cases α in $(d), \ldots, (j)$ does not hold.
By $(+)$ holds

$$\begin{aligned}
&(\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \\
&\quad \vee isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}}.time) \ ) \wedge \\
&ResMapCheck(\sigma_{\hat{n}}.res, v, m)
\end{aligned}$$

hence

$$\begin{aligned}
&r = \sigma_{\hat{n}}.res(v, src(m), m.id) \wedge \\
&\neg(0) \ r \neq \bot \wedge \\
&(1) \ r.path = m.path \wedge \\
&(2) \ r.ptr = m.ptr \wedge \\
&(3) \ r.first = m.first \wedge \\
&(4) \ r.last = m.last \wedge \\
&\neg(5) \ r.vrs \neq \emptyset \\
&\neg(6) \ r.vrs(m.idx) \neq \bot \wedge \\
&(7) \ r.vrs(m.idx).minBW = m.minBW \wedge \\
&(8) \ r.vrs(m.idx).maxBW = m.maxBW \wedge \\
&(9) \ r.vrs(m.idx).expT = m.expT
\end{aligned}$$

But this is in contradiction to the event's guard

$$ResMapCheck(\sigma_{\hat{n}}.res, v, \tilde{m})$$

(using $\tilde{v} = v$) and the case α that does not hold, i.e.

$$\begin{aligned}
&\sigma_{\hat{n}}.res(v, src(m), m.id).\alpha = m.\alpha \wedge \\
&\sigma_{\hat{n}}.res(v, src(\tilde{m}), \tilde{m}.id).\alpha = \tilde{m}.\alpha \wedge \\
&m.\alpha \neq \tilde{m}.\alpha
\end{aligned}$$

* $\tilde{v} \neq v$: The event's actions do not affect version of the reservation corresponding m, i.e.

$$\begin{aligned}
&\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx) \\
&\quad = \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)
\end{aligned}$$

and therefore the claim follows by $(+)$.

– $CMP(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in FWD with the following difference in case

$$\tilde{m} \approx m \wedge \tilde{m}.ptr = m.ptr \wedge \tilde{v} = v$$

We can show that

$$isRsvd(\sigma_{\hat{n}+1}.res, v, \tilde{m}, \sigma_{\hat{n}+1}.time)$$
$$ResMapCheck(\sigma_{\hat{n}+1}.res, v, \tilde{m})$$

instead of

$$isMrkd(\sigma_{\hat{n}+1}.res, v, \tilde{m}, \sigma_{\hat{n}+1}.time)$$
$$ResMapCheck(\sigma_{\hat{n}+1}.res, v, \tilde{m})$$

– $TRN(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in CMP.
– $UPT(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in CMP.
– $BWD(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in CMP.
– $FIN(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$:
– $CLT(\tilde{m},\tilde{m}',\tilde{a},\tilde{i})$: As in CRT_D.
– $ATK(\tilde{m},\tilde{v},\tilde{i})$: As in CRT_D.
– $RMV(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in RST it follows $(+)$, hence in particular

$$(\ isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time)$$
$$\vee\ isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time)\ ) \wedge$$
$$ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$

Two cases:

∗ $\tilde{m} \sim m \wedge \tilde{v} = v$: The event's action *remove* only sets the field *resBW* of the version corresponding to $\tilde{m}$ (and in this case m) to $\perp$, but keeps the other fields of $\sigma_{\hat{n}}.res$ the same. Hence it holds

$$isMrkd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time)$$
$$\Rightarrow isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time)$$
$$isRsvd(\sigma_{\hat{n}}.res, v, m, \sigma_{\hat{n}+1}.time)$$
$$\Rightarrow isMrkd(\sigma_{\hat{n}+1}.res, v, m, \sigma_{\hat{n}+1}.time)$$
$$ResMapCheck(\sigma_{\hat{n}}.res, v, m)$$
$$\Rightarrow ResMapCheck(\sigma_{\hat{n}+1}.res, v, m)$$

and therefore the claim follows by $(+)$.

∗ *Otherwise*: The event's actions do not affect version of the reservation corresponding m, i.e.

$$\sigma_{\hat{n}}.res(v, src(m), m.id).vrs(m.idx)$$
$$= \sigma_{\hat{n}+1}.res(v, src(m), m.id).vrs(m.idx)$$

and therefore the claim follows by $(+)$.

– $DST(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: As in RMV.
– $DRP(\tilde{m},\tilde{v},\tilde{i},\tilde{t})$: As in CRT_D.

□

Existence The following existence lemma shows that to any *FIN*-event at time t there was a *BWD* event before with an equivalent message $\bar{m}$ with the same *accBW* field at the previous AS $\bar{v}$ at time $\bar{t}$.

Lemma 5 (FIN-BWD-inductive-honest)**.**

$$\forall k \in \mathbb{N}$$
$$\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, e \in I, t \in \mathbb{N}.$$
$$\lambda_n = FIN(m, m', v, e, t) \wedge nodes(m) \subseteq H \wedge$$
$$0 \leq k < m.first \ \Rightarrow$$
$$\exists \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}.$$
$$\lambda_{\bar{n}} = BWD(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t}) \ \wedge$$
$$m \approx \bar{m}' \ \wedge\ \bar{m}'.ptr = k \ \wedge m.accBW = \bar{m}'.accBW$$

Proof. Induction on k.

- $k = 0$: Let $n \in \mathbb{N}$, $m, m' \in \mathscr{M}_R$, $v \in H$, $e \in I$, $t \in \mathbb{N}$ with $\lambda_n = FIN(m, m', v, e, t)$ and $nodes(m) \subseteq H$. ($0 \leq k < m.first$ holds in this case) We can show that

$$\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, e \in I, t \in \mathbb{N}.$$
$$\lambda_n = FIN(m, m', v, e, t) \wedge nodes(m) \subseteq H \wedge$$
$$0 \leq k < m.first \ \Rightarrow$$
$$\exists!\ \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}.$$
$$\lambda_{\bar{n}} = BWD(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t}) \ \wedge\ m = \bar{m}'$$

From $m = \bar{m}'$, hence $m \approx \bar{m}'$. We can also show that $m.ptr = \bar{m}'.ptr = 0$, i.e. $k = 0 = \bar{m}'.ptr$, and $m.accBW = \bar{m}'.accBW$.

- $k \to k+1$: By *IH* it holds

$$\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, e \in I, t \in \mathbb{N}.$$
$$\lambda_n = FIN(m, m', v, e, t) \wedge nodes(m) \subseteq H \wedge$$
$$0 \leq k < m.first \ \Rightarrow$$
$$\exists!\ \check{n} < n, \check{m}, \check{m}' \in \mathscr{M}_R, \check{v} \in H, \check{e} \in I, \check{t} \in \mathbb{N}.$$
$$\lambda_{\check{n}} = BWD(\check{m}, \check{m}', \check{v}, \check{e}, \check{t}) \ \wedge$$
$$m \approx \check{m}' \ \wedge\ \check{m}'.ptr = k \ \wedge m.accBW = \check{m}'.accBW$$

We need to show:

$$\exists!\ \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}.$$
$$\lambda_{\bar{n}} = BWD(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t}) \ \wedge$$
$$m \approx \bar{m}' \ \wedge\ \bar{m}'.ptr = k+1 \ \wedge m.accBW = \bar{m}'.accBW$$

Assume $n \in \mathbb{N}$, $m, m' \in \mathscr{M}_R$, $v \in H$, $e \in I$, $t \in \mathbb{N}$ with

$$\lambda_n = FIN(m, m', v, e, t)$$
$$nodes(m) \subseteq H$$
$$0 \leq k+1 < m.first.$$

From $0 \leq k+1 < m.first$ it follows that $0 \leq k < m.first$. Hence, by applying *IH* to k we get:

$$(*)\ \exists \check{n} < n, \check{m}, \check{m}' \in \mathscr{M}_R, \check{v} \in H, \check{e} \in I, \check{t} \in \mathbb{N}.$$
$$\lambda_{\check{n}} = BWD(\check{m}, \check{m}', \check{v}, \check{e}, \check{t}) \ \wedge$$
$$m \approx \check{m}' \ \wedge\ \check{m}'.ptr = k \ \wedge m.accBW = \check{m}'.accBW$$

and therefore we have

$$nodes(\check{m}) = nodes(m) \subseteq H$$
$$\lambda_{\check{n}} = BWD(\check{m}, \check{m}', \check{v}, \check{e}, \check{t})$$

We then show that

$$\begin{aligned}&\exists \hat{n} < \check{n}, \hat{m}, \hat{m}' \in \mathscr{M}_R, \hat{v} \in H, \hat{e} \in I, \hat{t} \in \mathbb{N}.\\ &\quad \lambda_{\hat{n}} = BWD(\hat{v}, \hat{e}, \hat{m}, \hat{m}', \hat{t}) \wedge\\ &\quad \check{m} = \hat{m}'\end{aligned}$$

Setting $\bar{n} := \hat{n}$ (i.e. $\bar{n} = \hat{n} < \check{n} < n$), $\bar{m} := \hat{m}$, $\bar{m}' := \hat{m}$, $\bar{v} := \hat{v}$, $\bar{e} := \hat{e}$, $\bar{t} := \hat{t}$ we get:

$$\begin{aligned}&\exists \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}.\\ &\quad \lambda_{\bar{n}} = BWD(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t})\end{aligned}$$

By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m} \approx \check{m}'$, and $m \approx \check{m}'$ we get:

$$m \approx \check{m}' \approx \check{m} = \hat{m}' = \bar{m}'$$

By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m}.ptr = \check{m}'.ptr + 1$ (by BWD event), and $\check{m}'.ptr = k$ (by $(*)$) we get:

$$\bar{m}'.ptr = \hat{m}'.ptr = \check{m}.ptr = \check{m}'.ptr + 1 = k + 1$$

By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m}.accBW = \check{m}'.accBW$ (by BWD event), and $m.accBW = \check{m}'.accBW$ (by $(*)$) we get:

$$\begin{aligned}m.accBW &= \check{m}'.accBW\\ &= \check{m}.accBW\\ &= \hat{m}'.accBW = \bar{m}'.accBW\end{aligned}$$

Together we get:

$$m \approx \bar{m}' \ \wedge \ \bar{m}'.ptr = k+1 \ \wedge m.accBW = \bar{m}'.accBW.$$

□

Uniqueness Each message processing event with message m a creates a new version of the reservation corresponding to m. Using uniqueness lemmata like the following, we can inductively show that this can only happen at most once, since otherwise there is a contradiction with Lemma 4 given before. We provide as representative uniqueness lemma the the following one for event *FIN*.

Lemma 6 (FIN-uniqueness)**.**

$$\begin{aligned}&\forall n_1, n_2 \in \mathbb{N}, m_1, m_1', m_2, m_2' \in \mathscr{M}_R, v_1, v_2 \in V, e_1, e_2 \in I, t_1, t_2 \in \mathbb{N}.\\ &\lambda_{n_1} = FIN(m_1, m_1', v_1, e_1, t_1) \wedge\\ &\lambda_{n_2} = FIN(m_2, m_2', v_2, e_2, t_2) \wedge\\ &m_1 \sim m_2 \ \wedge \ v_1 = v_2 \ \wedge\\ &nodes(m_1) \subseteq H \ \wedge\\ &n_1 < n_2 \ \wedge \ \sigma_{n_2}.time \leq m_1.expT\\ &\Rightarrow n_1 = n_2\end{aligned}$$

Proof. By $m_1 \sim m_2$, $m_1.ptr = m_2.ptr$, and $m_1.path = m_2.path$ it follows that $v := v_1 = v_2$ and $e := e_1 = e_2$. Together with assumption $nodes(m_1) \subseteq H$ we can establish that

$$\begin{aligned}(1)\quad &\exists \bar{n}_1 < n_1, \bar{m}_1, \bar{m}_1' \in \mathscr{M}_R, \bar{v}_1 \in H, \bar{e}_1 \in I, \bar{t}_1 \in \mathbb{N}.\\ &\quad \lambda_{\bar{n}_1} = BWD(\bar{m}_1, \bar{m}_1', \bar{v}_1, \bar{e}_1, \bar{t}_1) \wedge\\ &\quad m_1 = \bar{m}_1' \ \wedge\\ &\quad \forall \hat{n}_1.\ \bar{n}_1 < \hat{n}_1 \leq n_1 \Rightarrow m_1 \in \sigma_{\hat{n}_1}.buf(v, e)\end{aligned}$$

and similarly with $m_1.path = m_2.path$ and $nodes(m_2) \subseteq H$ follows (2) for n_2.

$$\begin{aligned}(2)\quad &\exists \bar{n}_2 < n_2, \bar{m}_2, \bar{m}_2' \in \mathscr{M}_R, \bar{v}_2 \in H, \bar{e}_2 \in I, \bar{t}_2 \in \mathbb{N}.\\ &\quad \lambda_{\bar{n}_2} = BWD(\bar{m}_2, \bar{m}_2', \bar{v}_2, \bar{e}_2, \bar{t}_2) \wedge\\ &\quad m_2 = \bar{m}_2' \ \wedge\\ &\quad \forall \hat{n}_2.\ \bar{n}_2 < \hat{n}_2 \leq n_2 \Rightarrow m_2 \in \sigma_{\hat{n}_2}.buf(v, e)\end{aligned}$$

By $m_1 \sim m_2$ and $m_1 = \bar{m}_1'$ and $\bar{m}_1' \approx \bar{m}_1$ (and analogously for $\bar{m}_2$), it follows that $\bar{m}_1 \sim \bar{m}_2$ Furthermore it follows:

$$\begin{aligned}nodes(\bar{m}_1) &=^{BWD(\bar{m}_1, \bar{m}_1', \ldots)} nodes(\bar{m}_1')\\ &=^{m_1 = \bar{m}_1'} nodes(m_1) \subseteq H\end{aligned}$$

and

$$\begin{aligned}\bar{m}_1.ptr &=^{BWD(\bar{m}_1, \bar{m}_1', \ldots)} \bar{m}_1'.ptr - 1\\ &=^{m_1 = \bar{m}_1'} m_1.ptr - 1\\ &=^{m_1.ptr = m_2.ptr} m_2.ptr - 1\\ &=^{m_2 = \bar{m}_2'} \bar{m}_2'.ptr - 1 =^{BWD(\bar{m}_2, \bar{m}_2', \ldots)} \bar{m}_2.ptr\end{aligned}$$

and

$$\begin{aligned}\bar{m}_1.path &=^{BWD(\bar{m}_1, \bar{m}_1', \ldots)} \bar{m}_1'.path\\ &=^{m_1 = \bar{m}_1'} m_1.path\\ &=^{m_1.path = m_2.path} m_2.path\\ &=^{m_2 = \bar{m}_2'} \bar{m}_2'.path =^{BWD(\bar{m}_2, \bar{m}_2', \ldots)} \bar{m}_2.path\end{aligned}$$

Three cases:

- $\bar{n}_1 < \bar{n}_2$. Then it holds:

$$\begin{aligned}\sigma_{\bar{n}_2}.time &\leq^{\bar{n}_2 < n_2} \sigma_{n_2}.time\\ &\leq^{\sigma_{n_2}.time \leq m_1.expT} m_1.expT\\ &=^{m_1 = \bar{m}_1'} \bar{m}_1'.expT\\ &=^{BWD(\bar{m}_1, \bar{m}_1', \ldots)} \bar{m}_1.expT\end{aligned}$$

  From this we can show that $\bar{n} := \bar{n}_1 = \bar{n}_2$.
- $\bar{n}_2 < \bar{n}_1$: Then it holds:

$$\begin{aligned}\sigma_{\bar{n}_1}.time &\leq^{\bar{n}_1 < n_1} \sigma_{n_1}.time\\ &\leq^{n_1 < n_2} \sigma_{n_2}.time\\ &\leq^{\sigma_{n_2}.time \leq m_2.expT} m_2.expT\\ &=^{m_2 = \bar{m}_2'} \bar{m}_2'.expT\\ &=^{BWD(\bar{m}_2, \bar{m}_2', \ldots)} \bar{m}_2.expT\end{aligned}$$

  With this we can show that

$$\begin{aligned}&\lambda_{\bar{n}_1} = BWD(\bar{m}_1, \bar{m}_1', \bar{v}_1, \bar{e}_1, \bar{t}_1) \wedge\\ &\lambda_{\bar{n}_2} = BWD(\bar{m}_2, \bar{m}_2', \bar{v}_2, \bar{e}_2, \bar{t}_2) \wedge\\ &\bar{m}_1 \sim \bar{m}_2 \ \wedge \ \bar{m}_1.ptr = \bar{m}_2.ptr \ \wedge \ \bar{m}_1.path = \bar{m}_2.path \ \wedge\\ &nodes(\bar{m}_2) \subseteq H \ \wedge\\ &\bar{n}_2 < \bar{n}_1 \ \wedge \ \sigma_{\bar{n}_1}.time \leq \bar{m}_2.expT\end{aligned}$$

  it follows $\bar{n} := \bar{n}_1 = \bar{n}_2$.

- $\bar{n}_1 = \bar{n}_2$: Hence, $\bar{n} := \bar{n}_1 = \bar{n}_2$ holds immediately.

Since π is a function and $\lambda_{\bar{n}_1} = \lambda_{\bar{n}_2}$ and therefore $\bar{m} := \bar{m}_1 = \bar{m}_2$, $\bar{m}' := \bar{m}'_1 = \bar{m}'_2$, $\bar{v} := \bar{v}_1 = \bar{v}_2$, $\bar{e} := \bar{e}_1 = \bar{e}_2$, and $\bar{t} := \bar{t}_1 = \bar{t}_2$. Set $m := m_1 = \bar{m}'_1 = \bar{m}'_2 = m_2$, $\bar{n} := \bar{n}_1 = \bar{n}_2$ in (2), then we get:

$$(2') \quad \forall \hat{n}.\ \bar{n} < \hat{n} \leq n_2 \Rightarrow m \in \sigma_{\hat{n}}.buf(v,e)$$

By assumption $\lambda_{n_1} = FIN(m_1, m'_1, v_1, e_1, t_1)$ and $m := m_1$ we get $m \notin \sigma_{n_1+1}.buf(v,e)$.

By $n_1 < n_2$ we get $\tilde{n} < n_1 + 1 \leq n_2$. By setting $\hat{n} := n_1 + 1$ in $(2')$ we get $m \in \sigma_{n_1+1}.buf(v,e)$, i.e. a contradiction. □

Using existence lemmata we can show inductively that there was an UPT even for any AS on $sgmt(m)$ and using corresponding uniqueness lemmata we can show that there was exactly one.

Lemma 7 (FIN-UPT-inductive-honest)**.**

$$\begin{aligned}
&\forall k \in \mathbb{N} \\
&\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, e \in I, t \in \mathbb{N}. \\
&\quad \lambda_n = FIN(m, m', v, e, t) \wedge nodes(m) \subseteq H \wedge \\
&\quad m.first \leq k < m.last \ \Rightarrow \\
&\quad \exists!\ \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}. \\
&\qquad \lambda_{\bar{n}} = UPT(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t}) \ \wedge \\
&\qquad m \approx \bar{m}' \ \wedge \ m.accBW = \bar{m}'.accBW
\end{aligned}$$

Proof. Induction on k.

- $k = m.first$: Let $n \in \mathbb{N}$, $m, m' \in \mathscr{M}_R$, $v \in H$, $e \in I$, $t \in \mathbb{N}$ with $\lambda_n = FIN(m, m', v, e, t)$ and $nodes(m) \subseteq H$ ($m.first \leq k < m.last$ holds in this case). We can show that

$$\begin{aligned}
(*)\ \exists!\ &\tilde{n} < n, \tilde{m}, \bar{m}' \in \mathscr{M}_R, \tilde{v} \in H, \tilde{e} \in I, \tilde{t} \in \mathbb{N}. \\
&\lambda_{\tilde{n}} = BWD(\tilde{m}, \bar{m}', \tilde{v}, \tilde{e}, \tilde{t}) \ \wedge \\
&m \approx \tilde{m}' \ \wedge \ m.accBW = \tilde{m}'.accBW
\end{aligned}$$

  Given

$$\begin{aligned}
&\lambda_{\tilde{n}} = BWD(\tilde{m}, \bar{m}', \tilde{v}, \tilde{e}, \tilde{t}) \\
&nodes(\tilde{m}) = nodes(m) \subseteq H
\end{aligned}$$

  we can then show that

$$\begin{aligned}
&\exists \bar{n} < \tilde{n}, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}. \\
&\quad \lambda_{\bar{n}} = UPT(\bar{v}, \bar{e}, \bar{m}, \bar{m}', \bar{t}) \ \wedge \\
&\quad \tilde{m} = \bar{m}'
\end{aligned}$$

  By $\tilde{m} = \bar{m}'$, $\tilde{m} \approx \tilde{m}'$ (by BWD event) and $m \approx \tilde{m}'$ (by $(*)$) it follows

$$m \approx \tilde{m}' \approx \tilde{m} = \bar{m}'$$

  By it follows:

$$\bar{m}'.ptr = k$$

  By $\tilde{m} = \bar{m}'$, $\tilde{m}.accBW = \tilde{m}'.accBW$ (by BWD event) and $m.accBW \approx \tilde{m}'.accBW$ (by $(*)$) it follows

$$m.accBW = \tilde{m}'.accBW = \tilde{m}.accBW = \bar{m}'.accBW$$

- $k \to k+1$: By IH it holds

$$\begin{aligned}
&\forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, e \in I, t \in \mathbb{N}. \\
&\quad \lambda_n = FIN(m, m', v, e, t) \wedge nodes(m) \subseteq H \wedge \\
&\quad m.first \leq k < m.last \ \Rightarrow \\
&\quad \exists!\ \check{n} < n, \check{m}, \check{m}' \in \mathscr{M}_R, \check{v} \in H, \check{e} \in I, \check{t} \in \mathbb{N}. \\
&\qquad \lambda_{\check{n}} = UPT(\check{m}, \check{m}', \check{v}, \check{e}, \check{t}) \ \wedge \\
&\qquad m \approx \check{m}' \ \wedge \ \check{m}'.ptr = k \ \wedge m.accBW = \check{m}'.accBW
\end{aligned}$$

  We need to show:

$$\begin{aligned}
&\exists!\ \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}. \\
&\quad \lambda_{\bar{n}} = UPT(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t}) \ \wedge \\
&\quad m \approx \bar{m}' \ \wedge \ \bar{m}'.ptr = k+1 \ \wedge m.accBW = \bar{m}'.accBW
\end{aligned}$$

  Assume $n \in \mathbb{N}$, $m, m' \in \mathscr{M}_R$, $v \in H$, $e \in I$, $t \in \mathbb{N}$ with

$$\begin{aligned}
&\lambda_n = FIN(m, m', v, e, t) \\
&nodes(m) \subseteq H \\
&m.first \leq k+1 < m.last.
\end{aligned}$$

  From $m.first \leq k+1 < m.last$ it follows that $0 \leq k < m.last$.
  Hence, by applying IH to k we get:

$$\begin{aligned}
(*)\ &\exists \check{n} < n, \check{m}, \check{m}' \in \mathscr{M}_R, \check{v} \in H, \check{e} \in I, \check{t} \in \mathbb{N}. \\
&\quad \lambda_{\check{n}} = UPT(\check{m}, \check{m}', \check{v}, \check{e}, \check{t}) \ \wedge \\
&\quad m \approx \check{m}' \ \wedge \ \check{m}'.ptr = k \ \wedge m.accBW = \check{m}'.accBW
\end{aligned}$$

  and therefore we have

$$\begin{aligned}
&nodes(\check{m}) = nodes(m) \subseteq H \\
&\lambda_{\check{n}} = UPT(\check{m}, \check{m}', \check{v}, \check{e}, \check{t}).
\end{aligned}$$

  It then follows that

$$\begin{aligned}
&\exists \hat{n} < \check{n}, \hat{m}, \hat{m}' \in \mathscr{M}_R, \hat{v} \in H, \hat{e} \in I, \hat{t} \in \mathbb{N}. \\
&\quad \lambda_{\hat{n}} = UPT(\hat{v}, \hat{e}, \hat{m}, \hat{m}', \hat{t}) \ \wedge \\
&\quad \check{m} = \hat{m}'
\end{aligned}$$

  Setting $\bar{n} := \hat{n}$ (i.e. $\bar{n} = \hat{n} < \check{n} < n$), $\bar{m} := \hat{m}$, $\bar{m}' := \hat{m}$, $\bar{v} := \hat{v}$, $\bar{e} := \hat{e}$, $\bar{t} := \hat{t}$ we get:

$$\begin{aligned}
&\exists \bar{n} < n, \bar{m}, \bar{m}' \in \mathscr{M}_R, \bar{v} \in H, \bar{e} \in I, \bar{t} \in \mathbb{N}. \\
&\quad \lambda_{\bar{n}} = BWD(\bar{m}, \bar{m}', \bar{v}, \bar{e}, \bar{t})
\end{aligned}$$

  By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m} \approx \check{m}'$, and $m \approx \check{m}'$ we get:

$$m \approx \check{m}' \approx \check{m} = \hat{m}' = \bar{m}'$$

  By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m}.ptr = \check{m}'.ptr + 1$ (by UPT event), and $\check{m}'.ptr = k$ (by $(*)$) we get:

$$\bar{m}'.ptr = \hat{m}'.ptr = \check{m}.ptr = \check{m}'.ptr + 1 = k+1$$

  By $\bar{m}' := \hat{m}'$, $\check{m} = \hat{m}'$, $\check{m}.accBW = \check{m}'.accBW$ (by UPT event), and $m.accBW = \check{m}'.accBW$ (by $(*)$) we get:

$$\begin{aligned}
m.accBW &= \check{m}'.accBW \\
&= \check{m}.accBW \\
&= \hat{m}'.accBW = \bar{m}'.accBW
\end{aligned}$$

Together we get:

$$m \approx \bar{m}' \ \wedge \ \bar{m}'.ptr = k+1 \ \wedge m.accBW = \bar{m}'.accBW$$

We can also show uniqueness, which establishes the lemma. □

Attacker Knowledge Given a reservation message m containing only honest ASes on its path, i.e. $nodes(m) \subseteq H$. Then it holds that this message can neither be contained in the attacker knowledge nor in any buffer of an attacker

Lemma 8 (Attacker-knowledge)**.**

$$\begin{aligned}&\forall m \in \mathscr{M}_R.\\ &\ nodes(m) \subseteq H \ \Rightarrow\\ &\ \forall n \in \mathbb{N}, a \in M, i \in I.\ m \notin \sigma_n.kwl \ \wedge \ m \notin \sigma_n.buf(a,i).\end{aligned}$$

Proof. We show by induction on $n \in \mathbb{N}$

$$\begin{aligned}&\forall n \in \mathbb{N}, a \in M, i \in I, m \in \mathscr{M}_R.\\ &\ nodes(m) \subseteq H \ \Rightarrow\\ &\quad m \notin \sigma_n.kwl \ \wedge \ m \notin \sigma_n.buf(a,i)\end{aligned}$$

i.e. the induction hypothesis *IH* for σ_n is given by

$$\begin{aligned}&\forall a \in M, i \in I, m \in \mathscr{M}_R.\\ &(*)\ nodes(m) \subseteq H \ \Rightarrow\\ &\quad (1)\ m \notin \sigma_n.kwl \ \wedge\\ &\quad (2)\ m \notin \sigma_n.buf(a,i)\end{aligned}$$

- $n = 0$: Since by definition of the initial state for any $m \in \sigma_0.kwl$ it holds that $src(m) \in A$ this leads to a contradictions with the assumption $nodes(m) \subseteq H$, i.e. (1) holds. By the definition of the initial state that $\forall v \in V, i \in I.\ \sigma_0.buf(v,i) = \emptyset$, the statement (2) holds trivially.
- $n \to n+1$: By case distinction on λ_n,
  - $CLT(\hat{m}, \hat{m}', \hat{a}, \hat{i}, \hat{t})$: Hence, $\hat{m} \approx \hat{m}'$, $\hat{m} \in \sigma_n.buf(\hat{a}, \hat{i})$, and $\hat{a} \in M$
    1) Assume $m \in \sigma_{n+1}.kwl = \sigma_n.kwl \cup \{\hat{m}'\}$. By case distinction:
       * $m \in \sigma_n.kwl \setminus \{\hat{m}'\}$: Contradiction to *IH* (1).
       * $m = \hat{m}'$: By guard of CLT it holds that $\hat{m} \in \sigma_n.buf(\hat{a}, \hat{i})$ and $\hat{m} \approx \hat{m}'$. By $m = \hat{m}'$ and $(*)$ follows, that $nodes(\hat{m}) = nodes(m) \subseteq= H$, which is in contradiction to *IH* (2) applied to $\hat{m}$
    2) Assume $\exists a \in M, i \in I.\ m \in \sigma_{n+1}.buf(a,i)$. Due to the action of *CLT* it holds that $\sigma_{n+1}.buf(a,i) = \sigma_n.buf(a,i)$. This is in contradiction to *IH* (2) applied to m.
  - $RES(\hat{m}, \hat{m}', \hat{v}, \hat{i}, \hat{t})$: Hence,

$$(+)\ \ \sigma_{n+1}.buf = \sigma_n.buf \begin{pmatrix} (\hat{v}, \hat{i}) & \mapsto & \sigma_n.buf(\hat{v}, \hat{i}) \setminus \{\hat{m}\} \\ (\hat{v}', \hat{i}') & \mapsto & \sigma_n.buf(\hat{v}', \hat{i}') \cup \{\hat{m}'\} \end{pmatrix}$$

with $\hat{v}' = \hat{m}'.path[\hat{m}'.ptr].as$.

1) Assume $m \in \sigma_{n+1}.kwl = \sigma_n.kwl$. By the action of *RES*, it holds that $m \in \sigma_{n+1}.kwl = \sigma_n.kwl$, which is in contradiction to *IH* (1) applied to m.
2) Assume $\exists a \in M, i \in I.\ m \in \sigma_{n+1}.buf(a,i)$ By case distinction due to $(+)$:
   * $m \in \sigma_n.buf(\hat{v}, \hat{i}) \setminus \{\hat{m}'\}$: Contradiction to *IH* (1).
   * $m = \hat{m}'$: By this it follows that $a = \hat{v}'$, hence, $a \in nodes(\hat{m}')$. Together with $\hat{m} \approx \hat{m}'$, hence $nodes(\hat{m}) = nodes(\hat{m}')$, and $(*)$ it follows, that

$$a = \hat{v}' \in nodes(\hat{m}) = nodes(m) \subseteq= H,$$

which is in contradiction to *IH* (2) applied to $\hat{m}$

- $CRT_R(\hat{m}, \hat{v}, \hat{i}, \hat{t})$: Due to the action of CRT_R, it holds

$$(+)\ \sigma_{n+1}.buf = \sigma_n.buf\left((\hat{v}, \hat{i}) \mapsto \sigma_n.buf(\hat{v}, \hat{i}) \cup \{\hat{m}\}\right)$$

1) Assume $m \in \sigma_{n+1}.kwl$. By the action of CRT_R, it holds that $m \in \sigma_{n+1}.kwl = \sigma_n.kwl$, which is in contradiction to *IH* (1) applied to m.
2) Assume $\exists a \in M, i \in I.\ m \in \sigma_{n+1}.buf(a,i)$. By case distinction on $(+)$ it follows:
   * $m \in \sigma_n.buf(\hat{v}, \hat{i}) \setminus \{\hat{m}\}$: By assumption $(\hat{v}, \hat{i}) = (a,i)$, which contradicts *IH* (2) for m.
   * $m = \hat{m}$: By this it follows that $src(m) = src(\hat{m})$. By the guards of CRT_R it follows that $src(\hat{m}) = \hat{v}$ and by $(+)$ and assumption it follows that $\hat{v} = a$, hence,

$$a = src(\hat{m}) = src(m) \in nodes(m) \subseteq H,$$

i.e. a contradiction to $(*)$ for m.

- $ATK(\hat{m}, \hat{v}, \hat{i}, \hat{t})$: Due to the action of ATK, it holds

$$(+)\ \sigma_{n+1}.buf = \sigma_n.buf\left((\hat{v}, \hat{i}) \mapsto \sigma_n.buf(\hat{v}, \hat{i}) \cup \{\hat{m}\}\right)$$

1) Assume $m \in \sigma_{n+1}.kwl$. By the action of ATK, it holds that $m \in \sigma_{n+1}.kwl = \sigma_n.kwl$, which contradicts *IH* (1) applied to m.
2) Assume $\exists a \in M, i \in I.\ m \in \sigma_{n+1}.buf(a,i)$. By case distinction on $(+)$ it follows:
   * $m \in \sigma_n.buf(\hat{v}, \hat{i}) \setminus \{\hat{m}\}$: By assumption $(\hat{v}, \hat{i}) = (a,i)$, which contradicts *IH* (2) for m.
   * $m = \hat{m}$: By this and the guard of ATK it follows that $m = \hat{m} \in \sigma_n.kwl$, which is in contradiction to *IH* (1) for m.

- *other* : Both fields *buf* and *kwl* stay unchanged.

□

3) Stability Theorem:

Theorem 3 (Stability Theorem)**.** If there are constant demands D between t_0 and t_1, then after time $t_0 + stabT$ all reservations allocate the ideal bandwidth until t_1, i.e.,

$$\begin{aligned}&\exists \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time = t_0 + stabT \ \wedge \\ &\forall m \in rng(D), r \in Res, n > \tilde{n}, v \in sgmt(m) \cap H. \\ &\quad \sigma_n.time \in \,]t_0 + stabT; t_1]\ \wedge\ r = \sigma_n.res_v(src(m), m.id) \\ &\quad\ \Rightarrow allocBW(r, \sigma_n.time) = \min_{x \in sgmt(m)}\{ideal(m, \sigma_{\tilde{n}}.res_x)\}\end{aligned}$$

Proof. We first show that after $maxT$ the requested demands in all reservation maps correspond to D,

$$\forall n \in \mathbb{N}. \sigma_n.time \in \,]t_0 + maxT; t_1].\ \sigma_n \vdash D$$

From this follows that for any honest AS v and reservation r corresponding to a reservation message $m \in rng(D)$ the function $demBW$ remains constant and evaluates to $m.maxBW$

$$\begin{aligned}&\forall n \in \mathbb{N}, v \in H, m \in rng(D), r \in Res. \\ &\quad \sigma_n.time \in \,]t_0 + maxT; t_1] \wedge r = \sigma_n.res_v(src(m), m.id) \\ &\quad\ \Rightarrow demBW(r,t) = m.maxBW.\end{aligned}$$

Since the *ideal* bandwidth computation on the first AS of a path only depends on the value of the function $demBW_v$ and not on the *ideal* bandwidth computations executed at previous ASes on the path, it remains constant as well. We can show by induction on the length of the message's path that the *ideal* computations for all ASes on the path remain constant after time $t_0 + stabT$.

Using this, we show that the result of the *avail* computation is greater than that of the *ideal* computation for every renewal

$$\begin{aligned}&\forall n \in \mathbb{N}, evt \in \{CMP, UPT, TRN\}. \\ &\quad \forall v \in H, m, m' \in \mathscr{M}_R, i \in I, t \in \mathbb{N} \\ &\quad\ \sigma_n.time \in \,]t_0 + stabT; t_1] \wedge \lambda_n = evt(m, m', v, i, t) \wedge m \vdash D \\ &\quad\quad \Rightarrow ideal(m, \sigma_n.res_v) \geq avail(m, \sigma_n.res_v).\end{aligned}$$

Hence, together with the definition of *finBW*

$$finBW(m) = \min\{m.maxBW, \min(m.accBW)\}$$

it follows that

$$allocBW(r, \sigma_n.time) = \min_{x \in sgmt(m)}\{ideal(m, \sigma_{\tilde{n}}.res_x)\}$$

as required. □

4) Local Properties of N-Tube's Computation: For a valid reservation message m, N-Tube's bandwidth allocation computation has the following four *local* properties: positivity, lower ideal bound, bounded-tube proportionality, and per-request proportionality. We define and prove these properties below. These properties hold at each AS, and are later used to prove the global properties (G1–G5).

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.

Definition (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

$$\textstyle\sum_{s' \in V} egDem(s', e) > \delta \cdot cap(x, e).$$

Analogous definitions apply to ingress links.

Positivity: The functions *avail* and *ideal* always compute strictly positive values.

Lemma 9 (Positivity)**.** For a valid request with $(*)$ $m.minBW = 0$ and $(+)$ $nodes(m) \subseteq H$ a positive amount of bandwidth is always allocated:

$$\begin{aligned}&\forall event \in \{CMP, TRN, UPT\}. \\ &\quad \forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, i \in I, t \in \mathbb{N}. \\ &\quad\ \lambda_n = event(m, m', v, i, t) \Rightarrow finBW(m, resM_v) > 0.\end{aligned}$$

Proof. W.l.o.g. we show the claim for the event UPT. For the events TRN and CMP the proof works analogously. By induction on n:

- $n = 0$: Since in a valid execution all buffers are empty in $\sigma_0.buf$ and therefore no message processing event can happen, the premise $\lambda_n = UPT(m, m', v, i, t)$ is not satisfied and the claim holds trivially.
- $n > 0$: By *IH* it holds

  $$\begin{aligned}&\forall n' < n, m, m' \in \mathscr{M}_R, v \in H, i \in I, t \in \mathbb{N}. \\ &\quad \lambda_{n'} = UPT(m, m', v, i, t) \Rightarrow finBW(m, resM_v) > 0.\end{aligned}$$

  Given AS v, message m with $(i, v, e) = cur(m)$. By the event's guard it holds that m is valid, in particular, that

  $$\begin{aligned}&(1)\quad m.maxBW > 0, \\ &(2)\quad cap(v,e) > 0, \\ &(3)\quad cap(v,i) > 0.\end{aligned}$$

  Furthermore, by the event's action it holds that

  $$\begin{aligned}&save(v \in V, resM \in ResMap, m' \in \mathscr{M}_R) = \\ &\quad \textbf{let} \\ &\quad\ (\!|\ i, v, e\ |\!) = cur(m) \\ &\quad\ finBW = \min(m'.accBW) \\ &\quad\ vrs' = (\!|\ minBW := m'.minBW; \\ &\qquad\qquad maxBW := m'.maxBW; \\ &\qquad\qquad idBW := \min\left(\delta cap(v,i), m'.maxBW, preIdBW(m')\right); \\ &\qquad\qquad resBW := finBW; \\ &\qquad\qquad expT := m'.expT\ |\!) \\ &\quad\ vrsM' = resM(v, src(m'), m'.id).vrs\left(m'.idx \mapsto vrs'\right) \\ &\quad\ res' = (\!|\ path := m'.path; \\ &\qquad\qquad ptr := m'.ptr; \\ &\qquad\qquad first := m'.first; \\ &\qquad\qquad last := m'.last; \\ &\qquad\qquad vrs := vrsM'\ |\!) \\ &\quad \textbf{in} \\ &\quad\ resM\left((v, src(m'), m'.id) \mapsto res'\right).\end{aligned}$$

and the event's guards *onPth*, i.e., $m.first < m.ptr < m.last$ and *ResMapCheck* it follows for the reservation corresponding to m, with
$r = \sigma_{n+1}.res(v, src(m), m.id)$, that

$$(4) \quad r.first < r.ptr < r.last$$
$$(5) \quad r.vrs(m.idx).expT > t$$

By the function *compute*

$$\begin{aligned} &compute(m \in \mathscr{M}_R, res \in ResMap, \delta \in]0;1[, t \in \mathbb{N}) = \\ &\quad \textbf{let} \\ &\quad newBW = (\!| \; avBW := avail(m, res, \delta, t); \\ &\qquad\qquad idBW := ideal(m, res, \delta, t) \; |\!) \\ &\quad \textbf{in} \\ &\quad m(\!| \; accBW := newBW \; \# \; m.accBW \; |\!). \end{aligned}$$

it follows that

$$\begin{aligned} &m'.accBW[m'.ptr] = \\ &\quad (\!| \; avBW := avail(m, res, \delta, t); idBW := ideal(m, res, \delta, t) \; |\!) \end{aligned}$$

Since $m'.maxBW = m.maxBW$ and (1) it suffices to show that both *avail* and *ideal* return a positive values.

- *avail* : By definition

$$\begin{aligned} &avail(m, resM, \delta, t) = \\ &\quad \textbf{let} \\ &\quad (\!| \; i, v, e \; |\!) = cur(m) \\ &\quad resM' = resM\,((v, src(m), m.id) \mapsto \bot) \\ &\quad resM'_v = filter(resM', v) \\ &\quad \textbf{in} \\ &\quad \delta \cdot \left(cap(v,e) - \sum_{\substack{r \in rng(resM'_v): \\ resEg(r) = e}} allocBW(r.vrs, t) \right) \end{aligned}$$

  By Lemma 10 it follows that

$$cap(v,e) > \sum_{\substack{r \in rng(resM'_v): \\ resEg(r) = e}} allocBW(r.vrs, t)$$

  Note, that in Lemma 10 the removal of all versions of reservation $(v, src(m), m.id)$

$$resM' = resM\,((v, src(m), m.id) \mapsto \bot)$$

  is not assumed. By $\delta > 0$ it follows that $avail(m, resM, \delta, t) > 0$.
- *ideal* : By definition it suffices to show that each of the three factors *reqRatio*, *linkRatio*, and *tubeRatio* is positive. Since their denominators are sums of non-negative summands it is sufficient to show that each nominator is positive. We show this only for the factor *reqRatio*, since the other cases can be shown with analogous arguments.

  The nominator of *reqRatio* (analogously for $reqRatio_{start}$)

$$\begin{aligned} &reqRatio_{transit}(v, s, id, i, resM, t) = \\ &\quad \frac{adjIdDem(v, resM(v, s, id), resM, t)}{transitDem(v, i, resM, t)} \end{aligned}$$

  is given by

$$\begin{aligned} &adjIdDem(v, r, resM, t) = \\ &\quad egScalFctr(v, s, e, resM, t) \cdot \\ &\quad \min\{cap(v,i), cap(v,e), idBW(r.vrs, t)\} \end{aligned}$$

  which by (2) and (3) is positive if *egScalFctr* and *idBW* are positive.

  $*$ *egScalFctr* : By the definition of *egScalFctr*

$$\begin{aligned} &egScalFctr(v, s, e, resM, t) = \\ &\quad \frac{\min(cap(v,e), egDem(v, s, e, resM, t))}{egDem(v, s, e, resM, t)} \end{aligned}$$

    and assumption (2) it suffices to show that $egDem(v, s, e, resM, t)$ is positive. By the definition of *egDem*

$$\begin{aligned} &egDem(v, s, e, resM, t) = \\ &\quad \sum_{\substack{r' \in rng(resM): \\ resSr(r') = s \\ resEg(r') = e}} reqDem(v, r', resIn(r'), e, t). \end{aligned}$$

    it suffices to show that $reqDem(v, r, i, e, t)$ is positive. By the definition of *reqDem*

$$\begin{aligned} &reqDem(v, r, i, e, t) = \\ &\quad \min\{cap(v,i), cap(v,e), demBW(r.vrs, t)\}. \end{aligned}$$

    and (2) and (3) it suffices to show that $demBW(r.vrs, t)$ is positive. By definition of *demBW*

$$\begin{aligned} &demBW(vrsM, t) = \\ &\quad \max\{vrs.maxBW \mid vrs \in rng(vrsM) \wedge \\ &\qquad vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\} \end{aligned}$$

    it suffices to show that

    (a) $r.vrs(m'.idx).maxBW > 0$
    (b) $r.vrs(m'.idx).minBW \leq r.vrs(m'.idx).resBW$
    (c) $r.vrs(m'.idx).expT \geq t$

    The fact (a) follows from (1).
    The fact (b) follows by assumption $(*)$ and that

$$r.vrs(m'.idx).minBW = m'.minBW = m.minBW = 0$$

    The fact (c) follows by (5), $n' < n$, and Lemma 3.

  $*$ *idBW* : By the definition of function *idBW*

$$\begin{aligned} &idBW(vrsM, t) = \\ &\quad \max\{vrs.idBW \mid vrs \in rng(vrsM) \wedge \\ &\qquad vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\}. \end{aligned}$$

it suffices to show that

(a) $r.vrs(m'.idx).idBW > 0$

(b) $r.vrs(m'.idx).minBW \leq r.vrs(m'.idx).resBW$

(c) $r.vrs(m'.idx).expT \geq t$

The fact (a) follows by the definition of the function *save*, in particular by

$$r.vrs(m'.idx).idBW := m'.accBW.[m'.ptr-1].idBW,$$

We can show using assumption $(+)$ that there is a $n' < n$ such that

$$\lambda_{n'} = UPT(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t})$$

and therefore

$$m'.accBW.[m'.ptr-1].idBW = ideal(\tilde{m}', \sigma_{n'}.res, \delta, \tilde{t})$$

By *IH* it follows that

$$finBW(\tilde{m}, resM_{\tilde{v}}) > 0$$

and therefore in particular

$$ideal(\tilde{m}', \sigma_{n'}.res, \delta, \tilde{t}) > 0$$

The facts (b) and (c) follow analogously to the case above.

Observe that all three factors contain their nominator as a summand in the denominator. Since all the denominators' summands are non-negative, it follows trivially that all three factors are less or equal than 1 and therefore

$$ideal(m, resM, \delta, t) \leq \delta \cdot cap(v,e).$$

□

Lemma 10 (Capacity-constraint-invariant)**.**

$$\begin{aligned}&\forall n \in \mathbb{N}, s, v \in H, e \in I, id \in \mathbb{N}, t \in \mathbb{N}.\\ &\quad resM'_v = filter(\sigma_n.res(v,s,id), v) \ \wedge \ cap(v,e) > 0\\ &\Rightarrow cap(v,e) > \sum_{\substack{r \in rng(resM'_v):\\ resEg(r)=e}} allocBW(r.vrs, t)\end{aligned}$$

Proof. By induction on n.

- $n = 0$: The inequality holds trivially since in the initial state $\sigma_0.res = \emptyset$, hence $allocBW(r.vrs,t) = 0$ for any $r \in rng(resM'_v)$. By assumption $cap(v,e) > 0$ the claim follows.
- $n \to n+1$: Assume $s, v \in H, e \in I, id \in \mathbb{N}, t \in \mathbb{N}$ with

  $$resM'_v = filter(\sigma_n(v,s,id).res, v) \wedge cap(v,e) > 0$$

  By *IH*

  $$\begin{aligned}&\forall s, v \in H, e \in I, id \in \mathbb{N}, t \in \mathbb{N}.\\ &\quad resM'_v = filter(\sigma_n(v,s,id).res, v) \wedge cap(v,e) > 0\\ &\Rightarrow cap(v,e) > \sum_{\substack{r \in rng(resM'_v):\\ resEg(r)=e}} allocBW(r.vrs, t)\end{aligned}$$

  By case distinction on λ_n. The relevant events are the following:

  – $FWD(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t})$: By the event's guard $onWay(m)$, it follows that the reservation that gets updated is filtered out by *filter*

  $$\begin{aligned}&filter(resM, v) =\\ &\ \lambda(s', id').\\ &\ \ \textbf{let } r = resM(v, s', id')\\ &\ \ \textbf{in } \big(\textbf{if } r.first \leq r.ptr \leq r.last \textbf{ then } resM(v, s', id') \textbf{ else } \bot\big)\end{aligned}$$

  Hence, $allocBW(r.vrs, t)$ stays the same the claim follows by *IH*.

  – $CMP(\tilde{m}, \tilde{m}', \tilde{v}, \tilde{i}, \tilde{t})$: The only relevant case is if $\tilde{v} = v$. Then by the event's action

  $$\begin{aligned}&save(v \in V, resM \in ResMap, m' \in \mathscr{M}_R) =\\ &\ \ \textbf{let}\\ &\ \ resM' = resM(v, src(m'), m'.id).vrs\big(m'.idx \mapsto vrs'\big)\\ &\ \ res' = (\!|\ path := m'.path;\\ &\qquad\quad ptr := m'.ptr;\\ &\qquad\quad first := m'.first;\\ &\qquad\quad last := m'.last;\\ &\qquad\quad vrs := vrsM'\ |\!)\\ &\ \ \textbf{in}\\ &\ \ resM\big((v, src(m'), m'.id) \mapsto res'\big).\end{aligned}$$

  and the event's guards *onPth*, i.e., $\tilde{m}.first < \tilde{m}.ptr < \tilde{m}.last$ and *ResMapCheck* it follows for the reservation corresponding to $\tilde{m}$, with $r = \sigma_{n+1}.res(v, src(\tilde{m}), \tilde{m}.id)$, that

  (4) $r.first < r.ptr < r.last$

  (5) $r.vrs(\tilde{m}.idx).expT > t$

  By *IH* it holds that

  $$\begin{aligned}&resM_v = filter(\sigma_n(v,s,id).res, v) \wedge cap(v,e) > 0\\ &\Rightarrow cap(v,e) > \sum_{\substack{\tilde{r} \in rng(resM_v):\\ resEg(\tilde{r})=e}} allocBW(\tilde{r}.vrs, t)\end{aligned}$$

  We need to show that

  $$\begin{aligned}&resM'_v = filter(\sigma_{n+1}(v,s,id).res, v) \wedge cap(v,e) > 0\\ &\Rightarrow cap(v,e) > \sum_{\substack{\tilde{r} \in rng(resM'_v):\\ resEg(\tilde{r})=e}} allocBW(\tilde{r}.vrs, t)\end{aligned}$$

The only reservation that changed from σ_n to σ_{n+1} is r to r', hence it holds (a)

$$\begin{aligned}
&\textstyle\sum_{\tilde{r}\in rng\left(resM'_v\right):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
&=\textstyle\sum_{\tilde{r}\in rng\left(resM'_v\right):\ \tilde{r}\neq r' resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r'.vrs,\tilde{t})\\
&=\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq rresEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r'.vrs,\tilde{t})
\end{aligned}$$

Furthermore, it holds that

$$(b)\quad r'.vrs(\tilde{m}.idx).resBW=\min\left(\tilde{m}'.maxBW,\min(\tilde{m}'.accBW)\right)$$

and for all other indices $idx\neq\tilde{m}.idx$ it holds that

$$(c)\quad r'.vrs(\tilde{m}.idx).resBW=r.vrs(\tilde{m}.idx).resBW$$

There are two cases:

* $r'.vrs(\tilde{m}.idx).resBW\ \leq\ allocBW(r.vrs,\tilde{t})$: From this together with (b) and (c) it follows

  $$(d)\ allocBW(r.vrs,\tilde{t})=allocBW(r'.vrs,\tilde{t}).$$

  From this it follows by (c) and *IH*

  $$\begin{aligned}
  &\textstyle\sum_{\tilde{r}\in rng\left(resM'_v\right):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
  &=^{(c)}\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r'.vrs,\tilde{t})\\
  &=^{(d)}\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r.vrs,\tilde{t})\\
  &=\textstyle\sum_{\tilde{r}\in rng(resM_v):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
  &<^{IH} cap(v,e)
  \end{aligned}$$

  and therefore the claim.
* $r'.vrs(\tilde{m}.idx).resBW>allocBW(r.vrs,\tilde{t})$: In this case by (b), (c), and the definition of *allocBW* it follows

  $$allocBW(r'.vrs,\tilde{t})=r'.vrs(\tilde{m}.idx).resBW$$

  and together with the definition of *finBW* it follows

  $$\begin{aligned}
  &r'.vrs(\tilde{m}.idx).resBW\\
  &\quad:=\min(\tilde{m}'.maxBW,\min(\tilde{m}'.accBW))\\
  &\quad\leq\tilde{m}'.accBW[\tilde{m}'.ptr].avBW\\
  &\quad=avail(\tilde{m},resM_v,\delta,\tilde{t})
  \end{aligned}$$

  hence, altogether it holds

  $$(e)\ allocBW(r'.vrs,\tilde{t})\leq avail(\tilde{m},resM_v,\delta,\tilde{t})$$

  By the definition of *avail*

  $$\begin{aligned}
  &avail(m,resM,\delta,t)=\\
  &\ \textbf{let}\\
  &\quad(\!|\ i,v,e\ |\!)=cur(m)\\
  &\quad resM'=resM\left((v,src(m),m.id)\mapsto\bot\right)\\
  &\quad resM'_v=filter(resM',v)\\
  &\ \textbf{in}\\
  &\quad\delta\cdot\left(cap(v,e)-\textstyle\sum_{r\in rng\left(resM'_v\right):\ resEg(r)=e} allocBW(r.vrs,t)\right)
  \end{aligned}$$

  Applying this to (c) and

  $$\begin{aligned}
  &\textstyle\sum_{\tilde{r}\in rng\left(resM'_v\right):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
  &=^{(c)}\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq rresEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r'.vrs,\tilde{t})\\
  &=^{(e)}\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+\\
  &\qquad avail(\tilde{m},resM_v,\delta,\tilde{t})\\
  &=^{(f)}\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+\\
  &\qquad\delta\left(cap(v,e)-\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\right)\\
  &=\delta cap(v,e)+(1-\delta)\textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
  &\leq\delta cap(v,e)+(1-\delta)\textstyle\sum_{\tilde{r}\in rng(resM_v):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
  &<^{IH}\delta cap(v,e)+(1-\delta)cap(v,e)=cap(v,e)
  \end{aligned}$$

  and the claim follows.

– $TRN(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: Analogous to CMP.
– $UPT(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: The only relevant case is if $\tilde{v}=v$. Then by the event's guard $isRsvd(\sigma_n.res,\tilde{v},\tilde{m},\tilde{t})$ it follows

  $$(11)\ r.vrs(\tilde{m}.idx).resBW\geq\min(\tilde{m}.accBW)$$

  Then by the event's action it follows for the updated reservation $r'=\sigma_{n+1}.res(v,src(\tilde{m}),\tilde{m}.id)$ that

  $$r'.vrs(\tilde{m}'.idx).resBW:=\min(\tilde{m}'.maxBW,\min(\tilde{m}'.accBW))$$

  and therefore

  $$r'.vrs(\tilde{m}'.idx).resBW\leq r.vrs(\tilde{m}'.idx).resBW$$

  From this and the fact that the version with index $(\tilde{m}'.idx$ is the only entry changed in the reservation map it follows

  $$(g)\quad allocBW(r'.vrs,\tilde{t})\leq allocBW(r.vrs,\tilde{t})$$

From this together with the *IV* it follows as in event CMP that

$$\begin{aligned}
&\textstyle\sum_{\tilde{r}\in rng(resM'_v):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
&=^{(c)} \textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r'.vrs,t)\\
&\leq^{(g)} \textstyle\sum_{\tilde{r}\in rng(resM_v):\ \tilde{r}\neq r\wedge resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)+allocBW(r.vrs,\tilde{t})\\
&= \textstyle\sum_{\tilde{r}\in rng(resM_v):\ resEg(\tilde{r})=e} allocBW(\tilde{r}.vrs,t)\\
&<^{IH} cap(v,e)
\end{aligned}$$

and therefore the claim.

- $BWD(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: Analogous to FWD.
- $FIN(\tilde{m},\tilde{m}',\tilde{v},\tilde{i},\tilde{t})$: Analogous to FWD.

In case of the other events, reservations get removed, hence,
$allocBW(r.vrs,t)$ stays the same or decreases for a corresponding reservation and the claim holds trivially.

□

Lower *ideal* Bound: Let m be a valid message with $(*)$ $m.minBW = 0$, $(+)$ $nodes(m) \subseteq H$, source s, ID id, and first AS v together with v's ingress and egress interface i_v and e_v. There is a strictly positive lower bound $G \cdot r_v \cdot m.maxBW$ on the *ideal* computation (even when all sources exceed their demands), where G only depends on m's path and $r_v = reqRatio(s,id,i_v,e_v)$ is the request ratio of m at v.

$$\begin{aligned}
&\forall event \in \{CMP, TRN, UPT\}.\\
&\ \forall n \in \mathbb{N}, m, m' \in \mathscr{M}_R, v \in H, i \in I, t \in \mathbb{N}.\\
&\ \ \lambda_n = event(m,m',v,i,t) \Rightarrow\\
&\ \ \exists G > 0. \forall x \in sgmt(m).\ ideal(m, resM_x) > G \cdot r_v \cdot m.maxBW
\end{aligned}$$

Proof. By assuming $(+)$ there is no attack AS on the path that can change the field *m.accBW* and a successful reservation done according to the N-Tube algorithm. By induction on the length of the path segment $sgmt(m)$ by the next Lemma 11. □

Lemma 11 (Ideal-computation-at-AS)**.**

$$\begin{aligned}
&\forall m \in \mathscr{M}_R, s \in H, v \in sgmt(m) \setminus \{first(m)\} \cap H.\\
&\ \forall i,e \in I, resM \in ResMap, t \in \mathbb{N}.\\
&\ \ cur(m) = v_i^e \ \wedge\ s = src(m) \ \wedge\\
&\ \ vrs = resM(v,src(m),m.id).vrs(m.idx) \ \wedge\\
&\ \ 0 < vrs.maxBW = m.maxBW \ \wedge\\
&\ \ m.accBW[m.first].idBW = m.maxBW \ \wedge\\
&\ \ vrs.idBW = m.accBW[m.ptr].idBW \ \wedge\\
&\ \ egDem(v,src(m),e,resM,t) \leq cap(v,e) \ \wedge\\
&\ \ inDem(v,src(m),i,resM,t) \leq cap(v,i) \ \wedge\\
&\ \ transitDem(v,i,resM,t) \leq cap(v,i)\\
&\ \ \Rightarrow \exists G_v > 0.\\
&\qquad ideal(m,resM,\delta,t) \geq G_v \cdot m.accBW[m.ptr].idBW
\end{aligned}$$

and for $v = first(m) \cap H$

$$\begin{aligned}
&\exists G_v > 0.\ ideal(m,resM,\delta,t) \geq\\
&\qquad G_v \cdot reqRatio_{start}(v,s,id,i,resM,t) \cdot m.maxBW
\end{aligned}$$

with

$$G_v := \frac{m.maxBW}{2 \cdot cap(v,i) \cdot \sum_{i' \in I_v} cap(v,i')} \cdot \delta \cdot cap(v,e).$$

Proof. Given m, $v \in sgmt(m)$, $i,e \in I$, $resM \in ResMap$, and $t \in \mathbb{N}$ with $cur(m) = v_i^e$, $src(m) = s$, and

$$\begin{aligned}
&(a)\ \ 0 < m.maxBW\\
&(b)\ \ egDem(v,src(m),e,resM,t) \leq cap(v,e)\\
&(c)\ \ inDem(v,src(m),i,resM,t) \leq cap(v,i)\\
&(d)\ \ transitDem(v,i,resM,t) \leq cap(v,i)\\
&(e)\ \ 0 < vrs.maxBW = m.maxBW\\
&(f)\ \ vrs.idBW = m.accBW[m.ptr].idBW\\
&(g)\ \ m.accBW[m.first].idBW = m.maxBW
\end{aligned}$$

By the definition of *ideal*

$$\begin{aligned}
&ideal(m,resM,\delta,t) =\\
&\ \textbf{let}\\
&\ \ (\!|\ i,v,e\ |\!) = cur(m)\\
&\ \ vrs' = (\!|\ minBW := m.minBW;\\
&\qquad\ maxBW := m.maxBW;\\
&\qquad\ idBW := \min(\delta cap(v,i), m.maxBW, preIdBW(m));\\
&\qquad\ resBW := m.minBW;\\
&\qquad\ expT := m.expT\ |\!)\\
&\ \ vrsM' = \emptyset\,(m.idx \mapsto vrs')\\
&\ \ res' = (\!|\ path := m.path;\\
&\qquad\ ptr := m.ptr;\\
&\qquad\ first := m.first;\\
&\qquad\ last := m.last;\\
&\qquad\ vrs := vrsM'\ |\!)\\
&\ \ resM' = resM\,((v,src(m),m.id) \mapsto res')\\
&\ \ resM'_v = filter(resM', v)\\
&\ \ tubeRatio = tubeRatio(v,i,e,resM'_v,t)\\
&\ \ \textbf{if}\ (m.first < m.ptr)\\
&\ \ \textbf{then}\ reqRatio = reqRatio_{transit}(v,src(m),m.id,i,resM'_v,t)\\
&\qquad\ linkRatio = linkRatio_{transit}(v,i,resM'_v,t)\\
&\ \ \textbf{else}\ reqRatio = reqRatio_{start}(v,src(m),m.id,i,resM'_v,t)\\
&\qquad\ linkRatio = linkRatio_{start}(v,i,resM'_v,t)\\
&\ \textbf{in}\\
&\ \ \min(\delta cap(v,i), m.maxBW,\\
&\qquad\ reqRatio \cdot linkRatio \cdot tubeRatio \cdot \delta \cdot cap(v,e)).
\end{aligned}$$

we consider the term:

$$reqRatio \cdot linkRatio \cdot tubeRatio(v,i,e,resM'_v,t) \cdot \delta \cdot cap(v,e)$$

We need to show that there are lower bounds for each of the following factors:

- *tubeRatio* : By its definition

$$tubeRatio(v,i,e,resM,t) = \frac{\min\{cap(v,i), tubeDem(v,i,e,resM,t)\}}{\sum_{i' \in I} \min\{cap(v,i'), tubeDem(v,i',e,resM,t)\}}$$

First we derive a lower bound for the fraction's nominator. By the definition of *tubeDem*

$$tubeDem(v,i,e,resM,t) = \sum_{\substack{r \in rng(resM): \\ resIn(r)=i \\ resEg(r)=e}} adjReqDem(v,r,i,e,resM,t)$$

A lower bound is the summand $r = resM(v,s,m.id)$

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

From (b) and the definition of *egScalFctr*

$$egScalFctr(v,s,e,resM,t) = \frac{\min(cap(v,e), egDem(v,s,e,resM,t))}{egDem(v,s,e,resM,t)}.$$

it follows that

$$(*)\ egScalFctr(v,s,e,resM,t) = 1$$

and the same for *inScalFctr* by (c). Hence, it is sufficient to provide a lower bound for *reqDem*

$$reqDem(v,r,i,e,t) = \min\{cap(v,i), cap(v,e), demBW(r.vrs,t)\}$$

By the definition of *egDem*

$$egDem(v,s,e,resM,t) = \sum_{\substack{r' \in rng(resM): \\ resSr(r')=s \\ resEg(r')=e}} reqDem(v,r',resIn(r'),e,t).r.path[r.ptr].inI,e).$$

and (b) it follows that

$$(A)\quad reqDem(v,r,i,e,t) \leq egDem(v,s,e,resM,t) \leq^{(b)} cap(v,e)$$

and analogously

$$reqDem(v,r,i,e,t) \leq inDem(v,s,i,resM,t) \leq^{(c)} cap(v,i)$$

it follows that

$$reqDem(v,r,i,e,t) = demBW(r.vrs,t).$$

By the definition of *demBW*

$$demBW(vrsM,t) = \max_{\substack{vrs \in \\ rng(vrsM)}} \{vrs.maxBW \mid vrs.minBW \leq vrs.resBW \wedge vrs.expT \geq t\}$$

it follows that

$$(B)\quad demBW(vrsM,t) \geq r.vrs(m.idx).maxBW = m.maxBW$$

All together we obtain that

$$\begin{aligned} &tubeRatio(v,i,e,resM,t) \\ &= \frac{\min\{cap(v,i), tubeDem(v,i,e,resM,t)\}}{\sum_{i' \in I} \min\{cap(v,i'), tubeDem(v,i',e,resM,t)\}} \\ &\geq \frac{\min\{cap(v,i), tubeDem(v,i,e,resM,t)\}}{\sum_{i' \in I_x} cap(v,i')} \\ &\geq^{(A),(B)} \frac{m.maxBW}{\sum_{i' \in I_v} cap(v,i')} \end{aligned}$$

with $I_v := \{i' \in I \mid cap(v,i') > 0\}$.

- *linkRatio* There are two cases $v = first(m)$ and $v \in sgmt(m) \setminus first(m)$. By the definition of $linkRatio_{transit}$

$$\begin{aligned} &linkRatio_{transit}(v,i,resM,t) = \\ &\textbf{let} \\ &\quad stDem = startDem(v,i,resM,t) \\ &\quad trDem = transitDem(v,i,resM,t) \\ &\textbf{in} \\ &\quad \frac{\min\{cap(v,i), trDem\}}{\min\{cap(v,i), stDem\} + \min\{cap(v,i), trDem\}}. \end{aligned}$$

By (d) it follows for its nominator

$$\begin{aligned} (D)\ &\min\{cap(v,i), trDem\} \\ &= \min\{cap(v,i), transitDem(v,i,resM,t)\} \\ &=^{(d)} transitDem(v,i,resM,t) \end{aligned}$$

and therefore

$$\begin{aligned} &linkRatio_{transit}(v,i,resM,t) \\ &= \frac{\min\{cap(v,i), trDem\}}{\min\{cap(v,i), stDem\} + \min\{cap(v,i), trDem\}} \\ &\geq \frac{\min\{cap(v,i), trDem\}}{2 \cdot cap(v,i)} \\ &=^{(D)} \frac{transitDem(v,i,resM,t)}{2 \cdot cap(v,i)} \end{aligned}$$

- $linkRatio_{start}$: In this case (D) does not hold, but by (C) and (g) it follows

$$\begin{aligned} (E)\ &adjIdDem(v,r,resM,t) \\ &\geq^{(C)} m.accBW[m.ptr].idBW \\ &=^{(g)} m.maxBW \end{aligned}$$

and we obtain as lower bound

$$\begin{aligned} &linkRatio_{start}(v,i,resM,t) \\ &= \frac{\min\{cap(v,i), stDem\}}{\min\{cap(v,i), stDem\} + \min\{cap(v,i), trDem\}} \\ &\geq \frac{\min\{cap(v,i), stDem\}}{2 \cdot cap(v,i)} \\ &\geq^{(E)} \frac{m.maxBW}{2 \cdot cap(v,i)} \end{aligned}$$

- $reqRatio_{transit}$: By the definition of *adjIdDem*

$$\begin{aligned}&adjIdDem(v,r,resM,t) = \\ &\quad egScalFctr(v,s,e,resM,t)\cdot \\ &\quad \min\{cap(v,i),cap(v,e),idBW(r.vrs,t)\}.\end{aligned}$$

and similar as above $(*)$ $egScalFctr(v,s,e,resM,t) = 1$ and $(+)$ $idBW(r.vrs,t) \leq cap(v,i)$, $cap(v,e)$ it follows that

$$\begin{aligned}(C)\ &adjIdDem(v,r,resM,t) \\ &=^{(+),(*)} idBW(r.vrs,t) \\ &\geq r.vrs(m.idx).idBW \\ &=^{(f)} m.accBW[m.ptr].idBW\end{aligned}$$

By (C) and (d) and the definition of $reqRatio_{transit}$

$$\begin{aligned}&reqRatio_{transit}(v,s,id,i,resM,t) \\ &\quad = \frac{adjIdDem(v,resM(v,s,id),resM,t)}{transitDem(v,i,resM,t)}\end{aligned}$$

it follows

$$\begin{aligned}&reqRatio_{transit}(v,s,id,i,resM,t) \\ &= \frac{adjIdDem(v,resM(v,s,id),resM,t)}{transitDem(v,i,resM,t)} \\ &\geq^{(C)} \frac{m.accBW[m.ptr].idBW}{transitDem(v,i,resM,t)}\end{aligned}$$

- $reqRatio_{start}$: By (E) and the definition of $reqRatio_{start}$ it follows

$$\begin{aligned}&reqRatio_{start}(v,s,id,i,resM,t) \\ &= \frac{adjIdDem(v,resM(v,s,id),resM,t)}{startDem(v,i,resM,t)} \\ &\geq^{(E)} \frac{m.maxBW}{startDem(v,i,resM,t)}\end{aligned}$$

Altogether we obtain the following lower bounds for *ideal* depending on the two cases $v = first(m)$ and $v \in sgmt(m) \setminus \{first(m)\}$, respectively:

- $v = first(m)$:

$$\begin{aligned}&ideal(m,resM,\delta,t) \\ &= reqRatio_{start}\cdot linkRatio_{start}\cdot tubeRatio\cdot\delta\cdot cap(v,e) \\ &\geq reqRatio_{start}\cdot\frac{m.maxBW}{2\cdot cap(v,i)}\cdot\frac{m.maxBW}{\sum_{i'\in I_v} cap(v,i')}\cdot\delta\cdot cap(v,e) \\ &= reqRatio_{start}\cdot m.maxBW\cdot\frac{m.maxBW}{2\cdot cap(v,i)\cdot\sum_{i'\in I_v} cap(v,i')} \\ &\quad\cdot\delta\cdot cap(v,e)\end{aligned}$$

Hence, we can set

$$G_v := \frac{m.maxBW}{2\cdot cap(v,i)\cdot\sum_{i'\in I_v} cap(v,i')}\cdot\delta\cdot cap(v,e)$$

which only depends on *m.maxBW* and the capacities on *m.path* of the network.

- $v \in sgmt(m) \setminus \{first(m)\}$:

$$\begin{aligned}&ideal(m,resM,\delta,t) \\ &= reqRatio_{transit}\cdot linkRatio_{transit}\cdot tubeRatio\cdot\delta\cdot cap(v,e) \\ &\geq \frac{m.accBW[m.ptr].idBW}{trDem}\cdot\frac{trDem}{2\cdot cap(v,i)}\cdot\frac{m.maxBW}{\sum_{i'\in I_v} cap(v,i')} \\ &\quad\cdot\delta\cdot cap(v,e) \\ &= m.accBW[m.ptr].idBW\cdot\frac{m.maxBW}{2\cdot cap(v,i)\cdot\sum_{i'\in I_v} cap(v,i')} \\ &\quad\cdot\delta\cdot cap(v,e)\end{aligned}$$

Hence, we can set

$$G_v := \frac{m.maxBW}{2\cdot cap(v,i)\cdot\sum_{i'\in I_v} cap(v,i')}\cdot\delta\cdot cap(v,e)$$

which only depends on *m.maxBW* and the capacities on *m.path* of the network.

□

Lemma 12 (Bounded Tube-Proportionality)**.** Provided that two ingress links i,i' of AS x are not congested, the *tubeRatio* computation splits the capacity of the egress link e proportionally according to the tube demands of i and i' to e

$$\frac{tubeRatio(i,e)}{tubeRatio(i',e)} = \frac{tubeDem(i,e)}{tubeDem(i',e)}.$$

In case i' is congested and its tube demand to e further increases, the ratio between both tube ratios remains fixed

$$\frac{tubeRatio(i,e)}{tubeRatio(i',e)} = \frac{tubeDem(i,e)}{cap(x,i')}.$$

Proof. Given two ingress links i,i' of AS x. If both ingress links i and i' are not congested, i.e.,

$$\begin{aligned}&(a)\quad \textstyle\sum_{\tilde{s}\in V} inDem(x,\tilde{s},i,resM,t) \leq cap(x,i) \\ &(b)\quad \textstyle\sum_{\tilde{s}\in V} inDem(x,\tilde{s},i',resM,t) \leq cap(x,i').\end{aligned}$$

By this it follows and the definition of *tubeDem* it follows

$$\begin{aligned}(a')\ &tubeDem(v,i,e,resM,t) \\ &= \sum_{\substack{r\in rng(resM):\\ resIn(r)=i\wedge resEg(r)=e}} adjReqDem(v,r,i,e,resM,t) \\ &\leq \sum_{\substack{r\in rng(resM):\\ resIn(r)=i\wedge resEg(r)=e}} reqDem(v,r,i,e,resM,t) \\ &\leq \sum_{\substack{r\in rng(resM):\\ resIn(r)=i}} reqDem(v,r,i,e,resM,t) \\ &= \sum_{\tilde{s}\in V}\sum_{\substack{r\in rng(resM):\\ resSr(r)=\tilde{s}\wedge resIn(r)=i}} reqDem(v,r,i,resEg(r),t) \\ &= \textstyle\sum_{\tilde{s}\in V} inDem(x,\tilde{s},i,resM,t) \\ &\leq^{(a)} cap(x,i)\end{aligned}$$

and similarly

$$(b')\ tubeDem(v,i',e,resM,t) \leq cap(x,i')$$

therefore it follows

$$\begin{aligned}
&\frac{tubeRatio(x,i,e,resM,t)}{tubeRatio(x,i',e,resM,t)}\\
&=\frac{\frac{\min\{cap(x,i),tubeDem(x,i,e,resM,t)\}}{\sum_{\tilde{i}\in I}\min\{cap(x,\tilde{i}),tubeDem(x,\tilde{i},e,resM,t)\}}}{\frac{\min\{cap(x,i'),tubeDem(x,i',e,resM,t)\}}{\sum_{\tilde{i}\in I}\min\{cap(x,\tilde{i}),tubeDem(x,\tilde{i},e,resM,t)\}}}\\
&=\frac{\min\{cap(x,i),tubeDem(x,i,e,resM,t)\}}{\min\{cap(x,i'),tubeDem(x,i',e,resM,t)\}}\\
&=^{(a'),(b')}\frac{tubeDem(x,i,e,resM,t)}{tubeDem(x,i',e,resM,t)}
\end{aligned}$$

Independent from ingress link i' being congestion, if i is not congested then it follows

$$\begin{aligned}
&\frac{tubeRatio(x,i,e,resM,t)}{tubeRatio(x,i',e,resM,t)}\\
&=\frac{\frac{\min\{cap(x,i),tubeDem(x,i,e,resM,t)\}}{\sum_{\tilde{i}\in I}\min\{cap(x,\tilde{i}),tubeDem(x,\tilde{i},e,resM,t)\}}}{\frac{\min\{cap(x,i'),tubeDem(x,i',e,resM,t)\}}{\sum_{\tilde{i}\in I}\min\{cap(x,\tilde{i}),tubeDem(x,\tilde{i},e,resM,t)\}}}\\
&=\frac{\min\{cap(x,i),tubeDem(x,i,e,resM,t)\}}{\min\{cap(x,i'),tubeDem(x,i',e,resM,t)\}}\\
&\geq^{(a')}\frac{tubeDem(x,i,e,resM,t)}{cap(x,i')}.
\end{aligned}$$

If the tube demand between i' and e exceeds $cap(x,i')$, i.e.,

$$tubeDem(x,i',e,resM,t)\geq cap(x,i')$$

then the last inequality becomes an equality

$$\frac{tubeRatio(x,i,e,resM,t)}{tubeRatio(x,i',e,resM,t)}=\frac{tubeDem(x,i,e,resM,t)}{cap(x,i')}$$

and stays fixed no matter how much $tubeDem(x,i',e,resM,t)$ increases. □

Per Request-Proportionality: Suppose two sources s and s' respectively make new reservations m and m' whose paths intersect on a connected segment $[v_0,\ldots,v_n]$, i.e.,

$$\begin{aligned}
&\exists n,k,k'\in\mathbb{N}.\\
&\ m.first<k\leq m.last\ \wedge\ m'.first<k'\leq m'.last\\
&\ \wedge\ \forall i\leq n.\ v_i=m.path[k+i].as=m.path[k'+i].as
\end{aligned}$$

If s and s' do not have excessive demands on this segment, then the ratio of their *ideal* bandwidth computations on the segment remain the same to their ratio at the first AS on the segment, even if links on the segment are congested

$$\begin{aligned}
&\forall v_i\in\{v_0,\ldots,v_n\}.\\
&\ \frac{ideal(m,resM_{v_i},\delta,t)}{ideal(m',resM_{v_i},\delta,t)}=\frac{ideal(m,resM_{v_0},\delta,t)}{ideal(m',resM_{v_0},\delta,t)}.
\end{aligned}$$

Lemma 13 (Per Request-Proportionality)**.**

$$\begin{aligned}
&\forall m,m'\in\mathscr{M}_R,r,r'\in Res,s,s'\in V,v\in H,i,e\in I,resM\in ResMap,t\in\mathbb{N}\\
&\ m.path[m.ptr]=m'.path[m'.ptr]=v_i^e\ \wedge\\
&\ s=src(m)\ \wedge\ s'=src(m')\ \wedge\\
&\ r=resM(v,s,m.id)\ \wedge\ r'=resM(v,s',m'.id)\ \wedge\\
&\ inDem(v,s,i,resM_v,t)\leq\delta cap(v,i)\ \wedge\\
&\ egDem(v,s',e,resM_v,t)\leq\delta cap(v,e)\\
&\ \Rightarrow\ \frac{ideal(m,resM,\delta,t)}{ideal(m',resM,\delta,t)}=\frac{idBW(r.vrs,t)}{idBW(r'.vrs,t)}
\end{aligned}$$

Proof. Given $m,m'\in\mathscr{M}_R$, $s,s'\in V$, $v\in H$, $i,e\in I$, $resM\in ResMap$, $t\in\mathbb{N}$ with

$$\begin{aligned}
&m.path[m.ptr]=m'.path[m'.ptr]=v_i^e\\
&s=src(m),s'=src(m')\\
&r=resM(v,s,m.id),r'=resM(v,s',m'.id)
\end{aligned}$$

and that s and s' have modest demands, i.e.,

$$\begin{aligned}
&(a)\quad inDem(v,s,i,resM_v,t),inDem(v,s',i,resM_v,t)\leq\delta cap(v,i)\\
&(b)\quad egDem(v,s,e,resM_v,t),egDem(v,s',e,resM_v,t)\leq\delta cap(v,e).
\end{aligned}$$

By this it follows that both *inScalFctr* and *egScalFctr* are equal to 1 and therefore (c) *adjIdDem* = *idBW* for m and m', respectively.

Using (a), (b), (c), and the definition of *ideal* it follows:

$$\begin{aligned}
&ideal(m,resM,\delta,t)\\
&=reqRatio_{transit}\cdot linkRatio_{transit}\cdot tubeRatio\cdot\delta\cdot cap(v,e)\\
&=\frac{adjIdDem(v,r,resM,t)}{trDem}\\
&\ \cdot\frac{\min\{\delta cap(v,i),trDem\}}{\min\{\delta cap(v,i),stDem\}+\min\{\delta cap(v,i),trDem\}}\\
&\ \cdot\frac{\min\{\delta cap(v,i),tubeDem(v,i,e,resM,t)\}}{\sum_{i'\in I}\min\{\delta cap(v,i'),tubeDem(v,i',e,resM,t)\}}\cdot\delta\cdot cap(v,e)\\
&=\frac{idBW(r.vrs,t)}{stDem+trDem}\\
&\ \cdot\frac{tubeDem(v,i,e,resM,t)}{\sum_{i'\in I}\min\{\delta cap(v,i'),tubeDem(v,i',e,resM,t)\}}\cdot\delta\cdot cap(v,e)
\end{aligned}$$

Due to the assumption $m.path[m.ptr]=m'.path[m'.ptr]=v_i^e$ it follows that *stDem*, *trDem*, and *tubeDem* are the same for m and m', and therefore the conclusion of this lemma. □

5) Global Properties of N-Tube: Given a valid execution, we formally specify the global properties (G1–G5). Table I shows a summary of these properties, which we define and prove in detail in the following.

Availability: If an honest AS makes a successful reservation m, then a positive amount of bandwidth, $finBW(m)$, will be reserved on its path until it expires.

Corollary 1 (Availability)**.** Assume s makes a successful reservation m at time t, then

$$\begin{aligned}
&\forall n\in\mathbb{N},v\in sgmt(m).\ \sigma_n.time\in\,]t;m.expT]\\
&\quad\Rightarrow\sigma_n.res_v(s,m.id).vrs(m.idx).resBW>0.
\end{aligned}$$

Proof sketch. Follows directly from Theorem 2 and the *positivity* property of the bandwidth computation from Section IV-D. □

Immutability: If an honest AS makes a successful reservation m, the reserved bandwidth stays the same for all ASes on m's path until it expires.

Corollary 2 (Immutability)**.** Assume s makes a successful reservation m at time t. Then

$$\begin{aligned}&\forall n,n' \in \mathbb{N}, v, v' \in sgmt(m).\ \sigma_n.time, \sigma_{n'}.time \in]t; m.expT] \\ &\quad \Rightarrow \sigma_n.res_v(s,m.id).vrs(m.idx).resBW \\ &\qquad = \sigma_{n'}.res_{v'}(s,m.id).vrs(m.idx).resBW.\end{aligned}$$

Proof sketch. This follows directly from the first statement in Theorem 2, since the *resBW* fields are set to $finBW(m)$ for all path ASes until m expires. □

Stability: During a period of constant demands, all reservations stabilize.

Corollary 3 (Stability)**.** Assume there are constant demands D between t_0 and t_1. Then

$$\begin{aligned}&\forall n,n' \in \mathbb{N}, r, r' \in Res, v \in H, m \in rng(D).\\ &\ \sigma_n.time, \sigma_{n'}.time \in]t_0 + stabT; t_1] \ \wedge \\ &\ r = \sigma_n.res_v(src(m), m.id) \ \wedge \ \ r' = \sigma_{n'}.res_v(src(m), m.id) \\ &\ \Rightarrow allocBW(r, \sigma_n.time) = allocBW\left(r', \sigma_{n'}.time\right)\end{aligned}$$

Proof sketch. This follows directly from Theorem 3. □

Minimum Bandwidth Guarantee: If there are constant demands D between t_0 and t_1, then there is a lower bound on the ideal bandwidth allocations that only depends on the request ratios on their first link and on the link capacities along their paths.

Corollary 4 (Minimum Bandwidth Guarantee)**.** Assume there are constant demands D between t_0 and t_1, then for any successful reservation request m there is a lower bound on the ideal bandwidth allocation that all ASes on m's path reserve until t_1.

$$\begin{aligned}&\exists \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time = t_0 + stabT \ \wedge \\ &\ \forall m \in rng(D), r \in Res.\ Succ(src(m), m, t_0) \\ &\ \Rightarrow \exists\ G > 0\ \forall n > \tilde{n}, v \in sgmt(m). \\ &\quad \sigma_n.time \in]t_0 + stabT; t_1] \ \wedge \ r = \sigma_n.res_v(src(m), m.id) \\ &\quad \Rightarrow allocBW(r, \sigma_n.time) \geq \\ &\qquad G \cdot reqRatio(m, \sigma_{\tilde{n}}.res_{first(m)}) \cdot m.maxBW\end{aligned}$$

Proof sketch. This follows directly from Theorem 3 and the *lower* ideal *bound* property of the *ideal* function in the bandwidth computation from Section IV-D. □

Bounded Tube Fairness: If there are constant demands D between t_0 and t_1, then, in the absence of congestion, bandwidth of egress links is allocated proportionally between tube demands and, in case some tube demands exceed their ingress links' capacities, their tube ratio is bounded.

Corollary 5 (Bounded Tube Fairness)**.** Assume there are constant demands D between t_0 and t_1, then

$$\begin{aligned}&\exists \tilde{n} \in \mathbb{N}.\ \sigma_{\tilde{n}}.time = t_0 + stabT \ \wedge \\ &\ \forall m \in rng(D), v \in sgmt(m) \cap H, i, i', e \in I, n > \tilde{n}. \\ &\ tubeDem_v(i,e) \in]0; \delta cap(v,i)] \wedge \\ &\ tubeDem_v(i',e) \in]0; \delta cap(v,i')] \\ &\quad \Rightarrow \frac{tubeRatio_v(i,e)}{tubeRatio_v(i',e)} = \frac{tubeDem_v(i,e)}{tubeDem_v(i',e)}.\end{aligned}$$

Analogously, in case $tubeDem_v(i',e) \geq cap(v,i')$, e.g., there are excessive demands from i' to e,

$$\frac{tubeRatio_v(i,e)}{tubeRatio_v(i',e)} = \frac{tubeDem_v(i,e)}{\delta cap(v,i')}.$$

Here $tubeRatio_v$ and $tubeDem_v$ denote the corresponding functions defined in Section IV-D that are computed at AS v.

Proof sketch. This follows directly from Theorem 3 and the *bounded tube-proportionality* property of the *ideal* function in the bandwidth computation from Section IV-D. □

Bounded tube-fairness implies that, when the system reaches a stable state, the *bounded tube-proportionality* property holds globally, i.e., on all links of honest ASes. This guarantees that in case of a link-flooding allocation attack the attacked ingress links' tube ratios are always bounded, which prevents that bandwidth reservations through the other ingress links will be reduced ad infinitum.

F. 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: $\mu = 0.0$, $\sigma = 1.0$