Network Neutrality Inference

Zhiyong Zhang\textsuperscript{1,2}, Ovidiu Mara\textsuperscript{2}, Katerina Argyraki\textsuperscript{2}

\textsuperscript{1}UESTC China, \textsuperscript{2}EPFL Switzerland

ABSTRACT
When can we reason about the neutrality of a network based on external observations? We prove conditions under which it is possible to (a) detect neutrality violations and (b) localize them to specific links, based on external observations. Our insight is that, when we make external observations from different vantage points, these will most likely be inconsistent with each other if the network is not neutral. Where existing tomographic techniques try to form solvable systems of equations to infer network properties, we try to form unsolvable systems that reveal neutrality violations. We present an algorithm that relies on this idea to identify sets of non-neutral links based on external observations, and we show, through network emulation, that it achieves good accuracy for a variety of network conditions.

Categories and Subject Descriptors
C.2.3 [Computer Communication Networks]: Network monitoring

Keywords
Network neutrality; network tomography

1. INTRODUCTION
Once a fundamental Internet property, network neutrality cannot be taken for granted today. There is evidence that Internet service providers (ISPs) differentiate against certain applications, typically BitTorrent [31], by deprioritizing [18], blocking [12], or shaping [11, 16] its traffic. More recently, there is evidence that ISPs are differentiating against traffic originating from specific content providers [1]. We are not saying that such differentiation should be illegal (although it is, in certain countries); our position is that—whether illegal or not—it should be transparent: if an ISP differentiates against specific end-hosts, protocols, or applications, that should be visible to the affected parties and regulators.

In this paper, we address two questions:
(a) Which neutrality violations can be detected based on external observations? There exist proposals for detecting specific kinds of neutrality violation. For instance, several systems can detect whether a network path differentiates based on transport- or application-layer information [18, 31, 15, 11]; however, they cannot detect differentiation based on source or destination IP address, even though end-users suspect that it does occur [1]. To the best of our knowledge, there exists no formal definition of which kinds of neutrality violation are detectable in the first place.
(b) Which neutrality violations can be localized to a specific link or sequence of links and how? Most existing systems can detect whether a network path violates neutrality, but they cannot localize it to a specific link or link sequence. An exception is NetPolice, which can determine whether a specific ISP violates neutrality by explicitly measuring the ISP’s performance for different traffic flows using traceroute probes [31]. To the best of our knowledge, there exists no system that can localize a neutrality violation to a specific link sequence without explicitly measuring its performance.

Our approach is inspired by network performance tomography, whose goal is to infer performance properties of links (loss rate, latency, congestion status, or congestion probability) based on external observations, i.e., without monitoring these links directly. A tomographic technique typically forms a system of equations

\[ \mathbf{y} = \mathbf{A} \cdot \mathbf{x}, \]

where $\mathbf{y}$ is a given vector of external observations (end-to-end path measurements), $\mathbf{A}$ is a given matrix that specifies the relationships between links and paths, and $\mathbf{x}$ is the vector of link properties that we are trying to infer. It then estimates $\mathbf{x}$, either by solving this system of equations (when it has a unique solution [7, 8, 6, 22, 21, 14]) or by picking a solution that has some desirable property, e.g., assumes the smallest number of problematic links [23, 13, 26, 10] or occurs with the highest probability [22]. This approach fundamentally relies on the assumption that the network is neutral (each link treats traffic from all paths the same), otherwise it would be impossible to express path measurements as a function of link properties and form a solvable system of equations.

Our insight is that, if the network is not neutral, when we make external observations from different vantage points, these will most likely be “inconsistent” with each other, i.e., any system of equations that we form based on them will be unsolvable. So, in a sense, we turn network performance tomography on its head: where existing tomographic techniques assume network neutrality and try to form solvable systems of equations to infer network properties, we try to form unsolvable systems that reveal neutrality violations. By applying this idea to carefully chosen “slices” of the net-
work, we can reason not only about the neutrality of the entire network, but also of link sequences or individual links.

One challenge in turning this insight into a practical algorithm is that, in practice, a network link that does not explicitly employ traffic differentiation (hence would be considered “neutral” by existing neutrality definitions) may behave “non-neutral,” e.g., it may appear to be congested to one traffic flow but non-congested to another, over the same time interval. Such network behavior may also lead to inconsistent external observations, and a naïve application of our theory would misinterpret these as willful neutrality violations. We present experimental evidence that we can avoid such misleading inconsistencies by comparing observations of similarly sized traffic aggregates.

After describing our terminology and notation (Section 2), we make the following contribution: we prove conditions under which a neutrality violation is “observable,” i.e., it manifests as a set of inconsistent external observations (Section 3), and a non-neutral link sequence is “identifiable,” i.e., it causes inconsistent external observations that cannot be attributed to any other link sequence (Section 4). We also present an algorithm that takes as input a network graph and external observations, and it identifies all the non-neutral link sequences that are identifiable (Section 5). Finally, we present an initial experimental evaluation of our algorithm on small topologies, based on an open-source network emulator [2], and we show that it achieves good accuracy in a variety of network conditions (Section 6). We close with a discussion of open issues (Section 7), related work (Section 8), and conclusions (Section 9).

2. SETUP

In this section, we present our goal (Section 2.1), assumptions (Section 2.2), and theoretical model (Section 2.3).

2.1 Goal

Our goal is to design an algorithm that takes as input a network graph and external observations (end-to-end measurements), and it identifies non-neutral link sequences. Given that certain neutrality violations are infeasible to detect (Section 3), our algorithm will suffer some false-negatives, but, ideally, it should not suffer false-positives, i.e., a neutral link sequence should never be incorrectly identified as non-neutral.

Neutrality violation is typically informally defined as differentiation based on flow type, i.e., the contents of the IP header, transport-layer header, and/or payload. For example, a network link may throttle traffic coming from a specific content provider (based on IP addresses) or from a specific peer-to-peer (P2P) network (based on port numbers or deep-packet inspection). When a network link throttles a traffic flow, it upper-bounds the rate at which the flow can send traffic, typically below the rate at which the flow generates traffic; this results in a higher packet-loss rate and/or latency than the one experienced by unthrottled traffic. So, traffic differentiation results in different traffic flows experiencing different performance when traversing the same link(s).

We define “neutrality violation” as the situation where traffic from two different network paths experiences different performance when traversing the same network link (formal definition in Section 2.3). Hence, in our model, a flow type is represented by a set of paths. For example, suppose a network link throttles traffic coming from a specific content provider; we model this by saying that the network has two “performance classes,” one class comprising all the paths that start at the content provider, and the other class comprising all the other paths. Similarly, suppose a network link throttles a specific kind of P2P traffic; again, we model this by saying that the network has two performance classes, one class comprising all the paths that carry this form of P2P traffic, and the other class comprising all the other paths. We will discuss the reason behind our choice of definition later in the paper, once we have presented our model and results.

We do not assume any knowledge on the network’s differentiation criteria, the number of performance classes it distinguishes, or which network paths belong to the same class. If one does have such knowledge, it is possible to detect and localize neutrality violations with a simpler approach than ours. For example, suppose two parties connected to the same ISP suspect that the ISP throttles their P2P traffic; to test this, they can exchange first P2P traffic, then some other kind of traffic, and compare the achieved performance. On the other hand, this approach does not work when (a) the ISP throttles all traffic between the two parties and/or (b) the two parties communicate over more than one ISPs (in general, administrative domains), and any one of them could be throttling their traffic.

2.2 Assumptions

We make three assumptions:

1. We assume the existence of a measurement platform and knowledge of the network graph that interconnects the measurement points. A link in the network graph may correspond to an IP-level link, a domain-level link, or, in general, a sequence of consecutive physical links.

2. We assume that the status of each network link is independent from the status of any other network link.

3. Theorem 1 (Section 3.3) and Lemma 3 (Section 4.2) assume that: within any given time interval, if a non-neutral network link introduces non-negligible packet loss in its top-priority performance class (formal definition in Section 2.3), it introduces non-negligible packet loss in the other performance classes as well.

Assumptions #2 and #3 simplify our analysis and algorithm, but they are not fundamental to our approach; we discuss how we plan to relax them in Section 7.
set of all links traversed by at least one path in pathset $p$ is the set of all links traversed by path $p$. Paths that traverse all the links in link sequence $l$ is the set of all paths in the network that are currently used.

We use $l$ to refer to some link in $L$, and $p_l$ to refer to the $k$-th link assuming an arbitrary ordering of all the $\mid L \mid$ links. We use $\lambda$ to refer to a loop-free sequence of consecutive links in $L$. We use $p$ to refer to some path in $P$, and $p_l$ to refer to the $i$-th path assuming an arbitrary ordering of all the $\mid P \mid$ paths.

A pathset is a set of paths, and we denote the set of all pathsets in the network with $P^*$ (the power set of $P$). We use $\pi$ to refer to some pathset. Given a set of pathsets $\Pi$, we use $\pi_l$ to refer to the $i$-th pathset assuming an arbitrary ordering of all the pathsets in $\Pi$.

We define the following helper functions: $\text{Paths}(l)$ is the set of all paths that traverse link $l$, $\text{Paths}(\lambda)$ is the set of all paths that traverse all the links in link sequence $\lambda$, $\text{Links}(p)$ is the set of all links traversed by path $p$, and $\text{Links}(\pi)$ is the set of all links traversed by at least one path in pathset $\pi$.

We say that link $l$ is distinguishable from link $l'$, when $\text{Paths}(l) \neq \text{Paths}(l')$.

### Performance classes and numbers.

A performance class is a set of paths (that, as we will see below, are treated the “same” by the network), and we denote the set of all performance classes by $C$. We use $c_n$ to refer to the $n$-th class assuming an arbitrary ordering of all the $\mid C \mid$ classes.

A link is characterized by a set of performance numbers $\{x(n) \mid n = 1..\mid C \mid\}$. When traffic from a path that belongs to the $n$-th class traverses this link, it experiences performance $x(n)$. We use $\{x_k(n) \mid n = 1..\mid C \mid\}$ to particularly denote the performance numbers of link $l_k$.

We say that a link is neutral when its performance number is the same for all performance classes: $x(n) = x, \forall n = 1..\mid C \mid$; otherwise, we say that the link is non-neutral. When there is only one performance class in the network, by definition, all links are neutral. We use $x_{nk}$ to particularly denote the performance number of neutral link $l_k$. We denote all the neutral links in the network by $L_n$, and all the non-neutral links in the network by $L_n$. The top-priority class of a non-neutral link is the class for which the link has the highest performance.

For example, in Figure 1(a), there are two performance classes, $\{p_1, p_3\}$ and $\{p_2\}$. Non-neutral link $l_1$ has performance numbers $\{x_1(1), x_1(2)\}$, while neutral link $l_3$ has performance number $x_3$. Traffic traversing $l_1$ experiences different performance, depending on which path it belongs to: traffic from path $p_1$ experiences performance $x_1(1)$, whereas traffic from path $p_2$ experiences performance $x_1(2)$. In contrast, all traffic traversing link $l_3$ experiences the same performance $x_3$.

Similarly, a link sequence is characterized by a set of performance numbers $\{x(n) \mid n = 1..\mid C \mid\}$.

Finally, a pathset $\pi$ is characterized by a performance number $y$. Given a set of pathsets $\Pi$, we use $y_l$ to denote the performance number of pathset $\pi_l$.

### Performance metrics.

In our experimental evaluation, we use the performance metric introduced in [22], which is defined as follows:

- Time is divided into intervals.
- We say that a link, link sequence, or path is congestion-free during a given time interval, when it introduces (or experiences, in the case of a path) negligible packet loss during that interval.
- The performance numbers of a link or link sequence $\lambda = \{x(n) \mid n = 1..\mid C \mid\}$ is the probability that $\lambda$ is congestion-free with respect to performance class $c_n$ during any given time interval.
- The performance number of pathset $\pi$ is $y \equiv \log(\hat{P}(\pi))$, where $P(\pi)$ is the probability that all the paths in $\pi$ are congestion-free during any given time interval.

In general, we can use any metric that is “additive” in the following sense:

1. The performance of a link sequence $\lambda$ for class $c_n$ is equal to the sum of the performance of its member links for that class:

$$\lambda(n) = \sum_{k \mid l_k \in \lambda} x_k(n).$$

For example, in Figure 1(a), link sequence $(l_1, l_3)$ has performance numbers $\lambda(n) = x_1(1) + x_3, \lambda(2) = x_1(2) + x_3$.

2. In a neutral network, the performance of a pathset $\pi$ is equal to the sum of the performance of its member links:

$$y = \sum_{k \mid l_k \in \text{Links}(\pi)} x_k.$$

For example, if the network in Figure 1(a) was neutral, pathset $\{p_1, p_2\}$ would have performance number $y = \sum_{k \mid l_k \in \text{Links}(\pi)} x_k$.
\[ x_1 + x_2 + x_3. \]

We restrict ourselves to performance metrics that satisfy both Equations 1 and 2, because we have found that these reveal the largest number of neutrality violations. Not all intuitive metrics fall into this category; we discuss how to address this limitation in Section 7.

**Definition of network neutrality.**

A neutral link sequence is a sequence of neutral links and a neutral network is one that has only neutral links. Conversely, a non-neutral sequence or network is one that includes at least one non-neutral link.

**Neutrality inference.**

The input to our problem consists of: the network \( G \) and the performance number of any pathset \( \pi \in P^* \). The desired output is the set of non-neutral links \( L_n \).

**Generalized routing matrix.**

A generalized routing matrix represents the relationships between the links \( L \) and a set of pathsets \( \Pi \) (Figure 1(b)). More formally: Given a set of pathsets \( \Pi \), a generalized routing matrix \( A(\Pi) \) is a \(|\Pi| \times |L|\) matrix with

\[
A_{ik} = \begin{cases} 
1, & \text{if at least one path in pathset } \pi_i \text{ traverses link } l_k \\
0, & \text{otherwise.}
\end{cases}
\]

**Systems of equations for a neutral network.**

Here are some of the equations we could write for the network in Figure 1, if it was neutral:

\[
\begin{align*}
\{p_1\} : & \quad y_1 = x_1 + x_2 \\
\{p_2\} : & \quad y_2 = x_1 + x_3 \\
\{p_1, p_2\} : & \quad y_3 = x_1 + x_2 + x_3.
\end{align*}
\]

Any such system of equations can be summarized as

\[ \vec{y} = A(\Pi) \cdot \vec{x}, \quad (3) \]

where \( \vec{x} = \{x_k \mid k = 1..|L|\} \) and \( \vec{y} = \{y_i \mid i = 1..|\Pi|\} \).

When the network is not neutral, the equations in System 3 are incorrect and—as we will see—the system is often unsolvable. This observation is the cornerstone of our work.

### 3. NETWORK NON-NEUTRALITY

In this section, we present a condition on the network \( G \) and the location of the non-neutral links \( L_n \), which is necessary and sufficient for observing non-neutrality using our approach. When a non-neutral network meets this condition, we can observe that it is non-neutral; when it does not meet this condition, it appears neutral to our approach. We first define “observability” (Section 3.1) and the “equivalent neutral network,” the basic construct that we use to formulate our result (Section 3.2), then state the condition and illustrate with examples (Section 3.3). The proof of the theorem is in the appendix.

#### 3.1 Definition of Observability

**Lemmas 1.** Consider a network with paths \( P \). If there exists a set of pathsets \( \Pi \subseteq P^* \) such that System 3 does not have a solution, then the network is non-neutral.

The proof is trivial: When all the links are neutral, the routing matrix correctly captures the relationships between the link performance numbers \( \vec{x} = \{x_k \mid k = 1..|L|\} \) and the external observations \( \vec{y} \), so System 3 has at least one solution.

Lemma 1 says that, if System 3 has no solution, the only possible explanation is that the network is non-neutral. For example, consider the network shown in Figure 1 and the system of equations

\[
\begin{align*}
\{p_1\} : & \quad y_1 = x_1 + x_2 \\
\{p_2\} : & \quad y_2 = x_1 + x_3 \\
\{p_3\} : & \quad y_3 = x_3 + x_4.
\end{align*}
\]

Suppose we observe that (a) \( y_1 = y_3 = 0 \) (paths \( p_1 \) and \( p_3 \) are always congestion-free), whereas (b) \( y_2 \neq 0 \) (path \( p_2 \) is occasionally congested). These two observations are inconsistent: (a) indicates that \( x_3 = 0 \) for all \( k \) (all the links are always congestion-free), whereas (b) indicates that either \( x_1 \neq 0 \) or \( x_3 \neq 0 \) (either \( l_1 \) or \( l_3 \) is occasionally congested). The only possible explanation for this inconsistency is that \( l_1 \) and/or \( l_3 \) are non-neutral and treat traffic from path \( p_2 \) worse than the other traffic.

**Definition 1.** Consider a non-neutral network with paths \( P \). We say that the network’s neutrality violation is observable, when there exists a set of pathsets \( \Pi \subseteq P^* \) such that System 3 does not have a solution.

There exist neutrality violations that are not observable with our approach. For example, consider the network shown in Figure 2(a), where link \( l_1 \) is non-neutral, treating traffic from path \( p_2 \) worse than traffic from path \( p_1 \). In this particular network, there exists no unsolvable System 3. The intuition is that the worse treatment that \( l_1 \) inflicts on path \( p_2 \) can always be attributed to link \( l_3 \). As a result, any set of external observations can be explained through a neutral behavior of the links, which means that we cannot detect neutrality violation based on external observations.

#### 3.2 Equivalent Neutral Network

From the point of view of the end-hosts, any non-neutral network with \(|L_n|\) neutral links, \(|L_n|\) non-neutral links, and \(|C|\) performance classes is equivalent to a neutral network with \(|L_n| + |L_n| \cdot |C|\) (neutral) links; by “equivalent” we mean that the two networks produce the same external observations. To generate an equivalent neutral network, we map each original non-neutral link \( l \) into \(|C|\) virtual neutral links, one of them modeling \( l \)’s common queue and the rest of them modeling \( l \)’s regulation of the lower-priority classes.
Figure 2: A non-neutral network (and its neutral equivalent) where neutrality violation is non-observable. There are two performance classes $C = \{ \{ p_1 \}, \{ p_2 \} \}$. Link $l_1$ is non-neutral: it treats traffic from $p_2$ worse than traffic from $p_1$.

Figure 2 shows an example: in the original network, non-neutral link $l_1$ has performance numbers $x_1(1)$ and $x_1(2)$; in the neutral equivalent, $l_1$ is mapped to two virtual links:

a) Virtual link $l_1^+(1)$ models $l_1$’s common queue. It has performance number $x_1(1)$, and it is traversed by both paths. It captures any bad performance that link $l_1$ may inflict on $p_1$, which will necessarily also be inflicted on $p_2$ (since it is lower-priority).

b) Virtual link $l_1^+(2)$ models $l_1$’s regulation of performance class $c_2$. It has performance number $x_1(2) - x_1(1)$, and it is traversed only by path $p_2$. It captures any extra bad performance that link $l_1$ may inflict on $p_2$ due to its regulation of performance class $c_2$.

An original non-neutral network $G = (V, L, P)$ and a neutral equivalent $G^+ = (V^+, L^+, P)$ are related as follows:

a) For each non-neutral link $l \in L_n$ with performance number $x$, there exists a link $l^+ \in L^+$ with the same performance number $x$ and $\text{Paths}(l^+) = \text{Paths}(l)$.

b) For each non-neutral link $l \in L_n$ with performance numbers $\{x(n) \mid n = 1..|C|\}$ and top-priority class $c_n$, there exist $|C|$ links $\{l^+(n) \mid n = 1..|C|\} \in L^+$, where:

i) $l^+(n^*)$ has performance number $x(n^*)$ and $\text{Paths}(l^+(n^*)) = \text{Paths}(l)$.

ii) $l^+(n), \forall n \neq n^*$ has performance number $x(n) - x(n^*)$ and $\text{Paths}(l^+(n)) = \text{Paths}(l) \cap c_n$.

As a second example, Figure 3(a) shows the neutral equivalent for the network in Figure 1(a): neutral link $l_2$ is mapped to virtual link $l_2^+$, while non-neutral link $l_1$ is mapped to $l_1^+(1)$ and $l_1^+(2)$.

Figure 3: Neutral equivalent for the network shown in Figure 1.

For any non-neutral network, there exists at least one neutral equivalent and potentially more. A neutral equivalent may include “loop links,” which start and end at the same node. For example, consider the network shown in Figure 4(a), where link $l_1$ and $l_2$ are non-neutral, treating traffic from path $p_1$ better than traffic from the rest. One of the neutral equivalents of this network (shown in Figure 4(b)) includes a loop link $l_1^+(2)$, which is traversed by $p_2$, $p_3$, and $p_4$. For this example, if we change the order of $l_1^+(1)$ and $l_1^+(2)$, we can get another neutral equivalent without loop link, shown in Figure 4(c). However, as the original network becomes larger and more complex, we may not build a loop-free neutral equivalent. Fortunately, it does not affect our following analysis.

We use $A^+$ to denote a generalized routing matrix of an equivalent neutral network. Clearly, even though there might exist more than one neutral equivalents. The routing matrix of any pathset in all the neutral equivalents are always identical. E.g., the neutral equivalents in Figure 4(b) and Figure 4(c) have the same routing matrix, shown in Figure 4(d).

3.3 Condition for Observability

THEOREM 1. Consider a non-neutral network with links $L$, and its equivalent neutral network with links $L^+$. The network’s neutrality violation is observable, if and only if there exists at least one virtual link $l^+(n) \in L^+$ that is distinguishable from any link in $L$.

We illustrate with three examples:

Non-observable violation.

Consider again the non-neutral network in Figure 2(a) and its neutral equivalent in Figure 2(b). In this case, none of the two virtual links in the neutral equivalent satisfies the con-
Observable violation #1.

Consider again the non-neutral network in Figure 1(a) and its neutral equivalent in Figure 3(a). In this case, virtual link \( l_1^+ (2) \) is distinguishable from any link in \( L \). Therefore, according to the theorem, this neutrality violation is observable. Indeed, we said earlier that \( l_1 \)’s effect on \( p_2 \) cannot be attributed to any set of neutral links (Section 3.1). Theorem 1 expresses this insight: if a virtual link \( l^+(n) \) in the neutral equivalent is distinguishable from any link in the original network, then the non-neutral behavior captured by \( l^+(n) \) cannot be masked, because it cannot be attributed to any other link(s).

Observable violation #2.

Consider the non-neutral network in Figure 5(a) and its neutral equivalent in Figure 5(b). Link \( l_1 \) introduces congestion into class-2 traffic with probability 0.5, while the rest of the network is congestion-free. In this case, virtual link \( l_1^+ (2) \) is distinguishable from any link in \( L \). Therefore, according to the theorem, this neutrality violation is observable.

At first, it may seem counter-intuitive that this neutrality violation is observable, as it looks similar to the one in Figure 2 (which is not). However, a closer look reveals that, in this case, there does exist an unsolvable system of equations:

If we monitor this network, we observe that:

\[
\begin{align*}
y_1 & = x_1 + x_2 \\
y_2 & = x_1 + x_3 \\
y_3 & = x_1 + x_4 \\
y_4 & = x_1 + x_3 + x_4.
\end{align*}
\]

These observations are inconsistent: (a) \( y_1 \) indicates that \( x_1 = x_2 = 0 \), which means that link \( l_1 \) is always congestion-free, (b) \( y_2, y_3, y_4 \) form a system with unique solution:

\[
\begin{align*}
x_1 & = \log(0.5) \\
x_3 & = 0 \\
x_4 & = 0
\end{align*}
\]

So, if we observe only path \( p_1 \), we conclude that link \( l_1 \) is always congestion-free, whereas if we observe only paths \( \{p_2, p_3\} \), their congestion can only be attributed to link \( l_1 \). The only explanation is that link \( l_1 \) is non-neutral and treats traffic from paths \( \{p_2, p_3\} \) worse than the rest.

As a side-note, this example illustrates the benefit of using performance metrics that can be defined and measured for pathsets, not only individual paths: The clue that gives away \( l_1 \)'s non-neutrality is the fact that \( p_2 \) and \( p_3 \) always experience congestion at the same time; assuming that link statuses are independent (Section 2.2), this correlation can only be attributed to non-neutral behavior by \( l_1 \). This clue emerges only if we observe \( p_2 \) and \( p_3 \) as a pair and measure the probability that they are both congestion-free. Theorem 1 guarantees that, as long as there exists a distinguishable link \( l^+(n) \) in the neutral equivalent, we can form an
observable neutrality violation of this slice can only be at-
apply Lemma 1 to a
network slice
multiple paths and can differentiate between them.
we only illustrate by example.
sequences, even individual links. For example, consider the

4. LINK NON-NEUTRALITY

In this section, we present a condition on the paths Paths(λ)
that traverse a link sequence λ, which is sufficient for cor-
rectly inferring λ’s neutrality. We first define the “network
slice,” the basic construct that we use to formulate our result
(Section 4.1), then state the condition and illustrate with ex-
amples (Section 4.2). The proofs of the lemmas are in the
appendix.

4.1 Network Slice

Lemma 1 can help us reason not only about the neutrality
of the entire network, but also about the neutrality of link
sequences, even individual links. For example, consider the
network shown in Figure 5(a), where the only link traversed
by multiple paths is l1. If we determine that this network is
non-neutral, then the only possible explanation is that link l1
is non-neutral, as it is the only link that handles traffic from
multiple paths and can differentiate between them.

To reason about the neutrality of a link sequence λ, we
apply Lemma 1 to a network slice $G_\lambda$, chosen such that any
observable neutrality violation of this slice can only be
attributed to λ. More specifically, we form a specialized
version of System 3:

$$\tilde{y} = A_\lambda (\Pi_\lambda) \cdot \tilde{x}. \quad (4)$$

We specify how we form this system in the appendix; here, we
only illustrate by example.

To reason about the neutrality of link l1 in Figure 4(a), we
form System 4 for $\lambda = \{l_1\}$ as follows:

a) We create a special set of pathsets $\Pi_{\{l_1\}}$ as follows:

i) We identify all path pairs whose only shared link
is $l_1$: $\{p_1, p_4\}$, $\{p_2, p_4\}$, and $\{p_3, p_4\}$.

ii) $\Pi_{\{l_1\}}$ consists of these path pairs plus their indi-
vidual paths: $\Pi_{\{l_1\}} = \{\{l_1\}, \{p_2\}, \{p_3\}, \{p_4\}\}$,
$\{p_1, p_4\}$, $\{p_2, p_4\}$, $\{p_3, p_4\}$.

b) We create network slice $G_{\{l_1\}}$ by abstracting away all
individual links in the network other than $l_1$, as shown
in Figure 6(a).

c) We form the system of equations that corresponds to
generalized routing matrix $A_{\{l_1\}} (\Pi_{\{l_1\}})$, shown in
Figure 6(b).

There are two key points about System 4: First, it typically
consists of a small number of equations, because $\Pi_\lambda$ tends
to be small (for any link sequence λ, there are typically few
path pairs whose only common link sequence is λ). Sec-
ond, once we have created $\Pi_\lambda$, the topology of the overall
network becomes irrelevant; the only factors that play a role
in System 4 are the performance numbers of the paths and
path pairs in $\Pi_\lambda$. This is different from existing tomography
techniques, which typically combine large numbers of path
measurements into a single large system of equations that is
determined by the topology of the entire network.

4.2 Identifiability of Non-neutral Links

**Lemma 2.** Consider a link sequence λ and its set of path-
sets $\Pi_\lambda$. If System 4 does not have a solution, then λ is non-
neutral.

System 4 is a special version of System 3 where the equa-
tions are picked for their particular relationship to link se-
quence λ. If System 4 does not have a solution, we al-
ready know from Lemma 1 that the network slice $G_\lambda$ is non-
neutral; Lemma 2 tells us that link sequence λ in particular
is non-neutral. For example, if the system in Figure 6(b)
does not have a solution, the only possible explanation is
that link $l_1$ is not neutral. The intuition is related to our discussion on Observable Violation #2 in Section 3.3: pathsets $\{p_1\}, \{p_1\}, \{p_1, p_4\}$ will yield one estimate for $l_1$’s performance; pathsets $\{p_2\}, \{p_1\}, \{p_2, p_4\}$ will yield a second estimate; these can only be different (and the system in Figure 6(b) unsolvable), if link $l_1$ treats a subset of the involved paths differently.

**Definition 2.** Consider a non-neutral link sequence $\lambda$ and its set of pathsets $\Pi_\lambda$. We say that $\lambda$ is identifiable, when System 4 does not have a solution.

There exist non-neutral links that are non-identifiable. For example, suppose we want to reason about the neutrality of link $l_2$ in Figure 4(a). To create $\Pi_\lambda$, we first identify every path pair whose only shared link is $l_2$. There are no such path pairs, $\Pi_\lambda = \emptyset$, and we cannot form System 4 for $\lambda = \{l_2\}$.

**Lemma 3.** Consider a non-neutral link sequence $\lambda$ with performance numbers $\{\hat{x}(n) \mid n = 1, \ldots, |C|\}$ and top-priority class $c_{n^\star}$. If the following conditions hold:

- there exist at least two path pairs $\pi_i, \pi_j$ in $\Pi_\lambda$
  \[(\exists \pi_i, \pi_j \in \Pi_\lambda);\]
- and a lower-priority class $c_n$ ($\exists n \neq n^\star \in C$);
- such that $\pi_i$ is entirely in class $c_n$ and $\pi_j$ is not
  \[(\pi_i \subseteq c_n, \pi_j \notin c_n);\]

then $\lambda$ is identifiable.

Informally, Lemma 3 says that a non-neutral link sequence is identifiable as long as it is traversed by a sufficiently diverse set of paths: We can determine that link sequence $\lambda$ is non-neutral, when it causes different path pairs in $\Pi_\lambda$ to experience inconsistent performance. This is guaranteed to happen for $l_1$ in Figure 4(a): on the one hand, path pair $\{p_2, p_4\}$ is entirely in performance class $c_2$, hence it will yield an estimate of $l_1$’s performance from the point of view of this performance class; on the other hand, path pair $\{p_1, p_4\}$ includes at least one path from performance class $c_1$, hence it will yield a different estimate of $l_1$’s performance. In contrast, there exist no path pairs at all that share only link $l_2$ in the same figure, hence we cannot identify it as a non-neutral link.

5. **Algorithm**

We will now present an algorithm that takes as input the network $G$ and the performance number of any pathset $\pi$, and it outputs a set of non-neutral link sequences $\Lambda_n$. We will use three metrics to characterize the quality of the algorithm:

**False-negative rate:** It is the fraction of non-neutral links that do not participate in any link sequence present in $\Lambda_n$.

**False-positive rate:** It is the fraction of neutral links that participate in neutral link sequences incorrectly present in $\Lambda_n$.

**Granularity:** It is the fraction of neutral links that do not participate in any link sequence present in $\Lambda_n$.

We move to set $\Lambda_n \setminus \{\lambda\}$, $\Lambda_n \leftarrow \Lambda_n \cup \{\lambda\}$

**Algorithm 1 Identification of non-neutral link sequences**

| Input: | The links L and paths P |
| Output: | The set of identif. non-neutral link seqs $\Lambda_n$ |
| 1: | $\Lambda_n \leftarrow \emptyset$, $\Lambda_n \leftarrow \emptyset$ |
| 2: | for each path pair $\{p_i, p_j\} \in P^*$ do |
| 3: | $\lambda = \text{Links}(p_i) \cap \text{Links}(p_j)$ |
| 4: | if $\lambda \notin \Lambda_n$ then |
| 5: | $\Lambda_n \leftarrow \Lambda_n \cup \{\lambda\}$, $\Pi_\lambda \leftarrow \emptyset$ |
| 6: | end if |
| 7: | $\Pi_\lambda \leftarrow \Pi_\lambda \cup \{p_i\}, \{p_j\}, \{p_i, p_j\}$ |
| 8: | end for |
| 9: | for each link sequence $\lambda \in \Lambda_n$ do |
| 10: | if $|\Pi_\lambda| \leq 5$ then |
| 11: | $\Lambda_n \leftarrow \Lambda_n \setminus \{\lambda\}$ |
| 12: | end if |
| 13: | if system $\tilde{y} = A_\lambda(\Pi_\lambda) \cdot \mathbf{x}$ has no solution then |
| 14: | $\Lambda_n \leftarrow \Lambda_n \setminus \{\lambda\}$, $\Lambda_n \leftarrow \Lambda_n \cup \{\lambda\}$ |
| 15: | end if |
| 16: | end for |
| 17: | return $\Lambda_n$ |

E.g., false-negative rate 10% means that 10% of the non-neutral links do not participate in any link sequence present in $\Lambda_n$.

Granularity is the average size of the link sequences in $\Lambda_n$. Smaller granularity indicates higher-quality results. The ideal is 1, which means that we can localize each observable neutrality violation to a single link.

False-positive rate: It is the fraction of neutral links that participate in neutral link sequences incorrectly present in $\Lambda_n$. E.g., false-positive rate 10% means that 10% of the neutral links participate in a neutral link sequence that is incorrect in $\Lambda_n$.

The core of our algorithm is Algorithm 1, which identifies all the non-neutral link sequences that are identifiable:

- Lines 1–12: We add to set $\Lambda_n$ every link sequence $\lambda$ for which $\Pi_\lambda$ contains at least 2 path pairs (which is equivalent to at least 5 pathsets).
- Lines 13–16: We move to set $\Lambda_n$ any link sequence $\lambda \in \Lambda_n$ for which System 4 has no solution.

If we assume no measurement errors, Algorithm 1 suffers 0 false-positives and potentially a few false-negatives corresponding to the non-identifiable non-neutral link sequences. This is because its output consists exactly of all the identifiable non-neutral link sequences: (a) Every link sequence $\lambda \in \Lambda_n$ is non-neutral. We have picked only link sequences for which System 4 does not have a solution. According to Lemma 2, any such link sequence is non-neutral. (b) Every non-neutral link sequence $\lambda \notin \Lambda_n$ is non-identifiable. We have discarded only link sequences for which System 4 does have a solution. By Definition 2, any such link sequence is non-identifiable.
For example, consider the network in Figure 4(a), and suppose both links $l_1$ and $l_2$ are non-neutral. In this case, there are two non-neutral link sequences that are identifiable $(\langle l_1, l_2 \rangle, \langle l_1, l_2 \rangle)$ and one non-neutral link sequence that is not $(\langle l_2 \rangle)$. The algorithm correctly identifies the former: At line 8, $\Lambda_n = \{ \langle l_1 \rangle, \langle l_1, l_2 \rangle \}$. Link sequence $\langle l_2 \rangle$ has not been added to $\Lambda_n$, because there exists no path pair such that $\text{Links}(p_i) \cap \text{Links}(p_j) = \{ \langle l_2 \rangle \}$. At line 16, $\Lambda_n = \{ \langle l_1 \rangle, \langle l_1, l_2 \rangle \}$, because both $\langle l_1 \rangle$ and $\langle l_1, l_2 \rangle$ allow the formation of an unsolvable System 4. Hence, in this example, the algorithm’s false-negative rate is 0 (both non-neutral links are present in $\Lambda_n$) and the false-positive rate is also 0. The granularity is 1.5 (the average length of the identified non-neutral link sequences), which reflects our uncertainty about the neutrality of link $l_2$.

After running Algorithm 1, we remove from $\Lambda_n$ all redundant link sequences. For example, suppose we have: $\Lambda_n = \{ \langle l_1, l_2 \rangle, \langle l_2, l_3 \rangle, \langle l_1, l_2, l_4 \rangle \}$; in this case, $\langle l_1, l_2, l_4 \rangle$ is redundant, because its presence in $\Lambda_n$ does not add new information about the neutrality of the network. Formally, we consider a link sequence $\lambda \in \Lambda_n$ redundant, if and only if:

$$\exists \{ \lambda_i | i = 1..m \} : \lambda_i \in \Lambda_n \cup \Lambda_n, \forall i \in [1, m]$$

$$\exists i \in [1, m] : \lambda_i \in \Lambda_n$$

$$\cup_{i=1}^{m} \lambda_i = \lambda.$$ 

In words, there exists a set of link sequences $\{ \lambda_i \}$ such that: all of them are either in $\Lambda_n$ or $\Lambda_n$, at least one of them is in $\Lambda_n$ (has been identified as non-neutral), and their union is equal to $\lambda$.

6. EXPERIMENTAL EVALUATION

In this section, we first describe our network emulator (Section 6.1) and how we process the measurements that we collect from it (Section 6.2). Then we present experimental results from two small topologies (Section 6.3 and Section 6.4) and close with take-away points (Section 6.5).

6.1 Network Emulator

We perform our experiments within an open-source network emulator, where end-hosts generate actual TCP traffic and network links implement actual traffic-differentiation mechanisms. We do not simulate packet loss or queuing delay: traffic experiences actual packet loss and queuing delay depending on the queuing policies and the level of congestion it encounters in the network. We do not assume noise-free end-to-end measurements: end-hosts measure the performance of end-to-end paths based on the actual traffic they exchange.

Our emulator is similar to ModelNet [29], with the difference that the network is emulated by a user-level process, not a kernel module. We opted for a user-level implementation, because we found it easier to debug, and it avoids the overhead of porting between different operating systems. As in ModelNet, the role of end-hosts is played by virtual network interfaces that exchange real traffic. The role of the network (the relays) is played by a network process that implements network queues and policies. The size of each queue is set according to the maximum round-trip time (RTT) experienced by traffic traversing the queue. The only aspect of the network that is simulated is the propagation delay of links.

We implemented two traffic-differentiation mechanisms that are deployed in current network devices: policing and shaping. Both of them limit the fraction of a link’s capacity that is consumed by a given performance class. Policing relies on a token bucket; the rate at which tokens are added to the bucket determines the maximum rate of the targeted performance class; the size of the bucket determines the maximum allowed burst; any excess traffic (that does not fit in the bucket) is immediately dropped. Shaping is similar, with the difference that any excess traffic is buffered in a dedicated queue.

In the experiments presented in this paper, the network either is neutral or implements $|C| = 2$ performance classes. When we say that a network link “implements policing,” we mean that it passes all class-2 traffic through one policer, whose rate varies across experiments from 50% to 20% of link capacity. When we say that a network link “implements shaping,” we mean that it passes all class-2 traffic through one shaper, whose rate $R$ varies across experiments from 50% to 20% of link capacity, while it passes all class-1 traffic through another shaper with rate $1 - R$. Unless otherwise stated, link capacity and policing/shaping rate take the default values shown in Table 1.

In each experiment, the end-hosts generate TCP traffic for 10 minutes. Each pair of communicating end-hosts starts a number of parallel TCP flows with the transfer size following a Pareto distribution; when a TCP flow ends, a new one starts after an idle time that is governed by an exponential distribution. We chose this model, because there is evidence that it captures well the communication between pairs of Internet end-hosts [9], but it is not crucial to our results—it is just one way of generating dynamic traffic patterns. Across experiments, we vary the number of parallel flows per path, as well as the parameters of the Pareto and exponential distributions (that govern flow size and inter-flow idle time, respectively). Unless otherwise stated, these parameters take the default values shown in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottleneck capacity (Mbps)</td>
<td>100</td>
</tr>
<tr>
<td>RTT (ms)</td>
<td>50, 80, 120, 200</td>
</tr>
<tr>
<td>Policing/shaping rate (%)</td>
<td>20, 30, 40, 50</td>
</tr>
<tr>
<td>Congestion-control algorithm</td>
<td>CUBIC, NewReno</td>
</tr>
<tr>
<td>Parallel TCP flows per path</td>
<td>1, 2, 15, 20, 70</td>
</tr>
<tr>
<td>Mean TCP flow size (MB)</td>
<td>1, 10, 40, 10000</td>
</tr>
<tr>
<td>Mean inter-flow gap (s)</td>
<td>10</td>
</tr>
<tr>
<td>Loss threshold (%)</td>
<td>1, 5, 10</td>
</tr>
<tr>
<td>Measurement interval (ms)</td>
<td>100, 200, 500</td>
</tr>
</tbody>
</table>

Table 1: Experiment parameters. Default values are in bold.
6.2 Measurement Processing

By our definition of neutrality, a neutral link \( l \) inflicts congestion on any path \( p \in \text{Paths}(l) \) with the same probability. In practice, we found that this may not hold, because packet loss is not uniform: if path \( p_2 \) carries more and/or larger TCP flows than path \( p_1 \), the same neutral link \( l \) may drop a different fraction of packets from \( p_2 \) than \( p_1 \) during each time interval. As a result, even a neutral link may have different congestion probabilities for different performance classes.

To account for the above, we normalize our path measurements, such that they refer to traffic aggregates of the same rate. In particular, when we form System 4 for link sequence \( \lambda \), we create the vector \( \vec{y} \) as follows: (a) We discount certain packets from our measurements, such that: in each time interval, all paths in \( \text{Paths}(\lambda) \) appear to have sent the same number of packets. (b) In each time interval, for each path \( p \in \text{Paths}(\lambda) \), we count the fraction of packets that were lost; if this fraction is below a loss threshold, we decide that \( p \) was congestion-free in this interval. (c) In each time interval, for each pathset \( \pi \in \Pi_\lambda \), we decide that \( \pi \) was congestion-free in this interval, when all its member paths were congestion-free. (d) We compute \( P(\pi) \) as the fraction of intervals in which \( \pi \) was congestion-free. The exact process by which we create \( \vec{y} \) is stated as Algorithm 2 in the appendix.

A key step of our algorithm is to determine whether various instances of System 4 “have a solution” (line 13 of Algorithm 1). In practice, none of these systems has a perfect solution, but some are significantly “more unsolvable” than others. We decide whether System 4 for link sequence \( \lambda \) “has a solution” as follows: (a) We estimate \( \lambda \)’s performance number based on each path pair in \( \Pi_\lambda \). We compute the system’s unsolvability as the absolute difference between the minimum and the maximum estimate. (b) Based on this unsolvability, we assign the system to one of two clusters using standard clustering; we decide that the system “has a solution,” when it belongs to the low-unsolvability cluster.

6.3 Results: Single Shared Link

We first consider a Dumbbell topology with a single shared link that (in some experiments) implements traffic differentiation. This could correspond to the scenario in Figure 7(a), where an ISP throttles all traffic from servers \( S_3 \) and \( S_4 \). In this case, the topology in Figure 7(b) is the network slice that we need to monitor in order to reason about the neutrality of the shared link \( l_5 \).

Experimental setup.

We present nine experiment sets, summarized in Table 2. We always refer to pathset \( \{p_1, p_2\} \) as “class \( c_1 \)” and to pathset \( \{p_3, p_4\} \) as “class \( c_2 \).” In experiments where the network is neutral, \( c_1 \) and \( c_2 \) do not constitute, strictly speaking, different performance classes, but we refer to them this way for simplicity.

In the first three experiment sets, the shared link does not implement any traffic differentiation. To make things difficult for our algorithm, we try to create network conditions that could be misinterpreted as non-neutrality. For instance, in experiment set #1, class \( c_1 \)’s average flow size is 1Mb, while class \( c_2 \)’s average flow size varies from 1Mb to 10Gb (a different value per experiment). Similarly, in sets #2 and #3, the two classes have different RTTs or use different congestion-control algorithms. As a result, in several of these experiments, the two classes exhibit dramatically different behavior, for instance, one class spends significantly more time in TCP slow start than the other.

In the last six experiment sets, the shared link polices or shapes the traffic in class \( c_2 \). Here, the difficult scenarios for our algorithm are the ones where all the paths carry the same kind of traffic. In each experiment set, we vary the average flow size, RTT, or policing/shaping rate of the shared link across experiments; however, in any single experiment, class-\( c_1 \) traffic and class-\( c_2 \) traffic have the same average flow size, RTT, and congestion-control algorithm.

Results.

In all these experiments, our algorithm correctly decides whether the shared link is neutral or not.

Figure 8 shows the external observations for each experiment set. In each graph, the \( y \)-axis represents the probability that a path is congested, while the \( x \)-axis represents different experiments. For each \( x \)-axis value (each experiment), there are four data points, one for each of the four paths in our topology. For instance, in experiment set #1, when class \( c_2 \) has mean flow size 40Mb, each of the four paths is congested with probability 40% (Figure 8(a), third set of bars).

Figure 8 provides insight into how the algorithm works: When the shared link does not implement traffic differen-
tion, the 4 paths are congested with the same probability (top three graphs in Figure 8), which leads to consistent observations. When the shared link implements policing or shaping, the two paths in class \(c_2\) are congested significantly more often than the two paths in class \(c_1\) (bottom six graphs in Figure 8), which leads to inconsistent observations. One exception is that when the shaping rate is 0.5 (first experiment in Figure 8(i)), in which case, the link throttles both traffic classes at exactly the same rate, and thus the 4 paths are congested with the same probability.

### 6.4 Results: Multiple Shared Links

Next, we consider the topology in Figure 9, which has multiple shared links. Links \(l_5\), \(l_{14}\), and \(l_{20}\) implement policing, while the other links do not implement explicit traffic differentiation. This could correspond to the scenario where routers \(R_1\) to \(R_5\) form the backbone of a tier-1 ISP, while each of routers \(R_6\), \(R_7\), \(R_{10}\), \(R_{11}\), and \(R_{12}\) belongs to a different tier-2 ISP or content provider. The tier-1 ISP uses policing on links \(l_{14}\) and \(l_{20}\) to throttle video or P2P traffic entering its network from routers \(l_7\) and \(l_{11}\), respectively; it uses policing on link \(l_5\) to prevent internal P2P traffic from overloading its backbone network. We do not intend this to be a realistic topology, but to create a challenging scenario for our algorithm, with multiple shared links and bottlenecks.

**Experimental setup.**

We present an experiment where the network differentiates against long flows. There are three types of end-hosts: dark gray end-hosts with lines exchange long flows, light gray end-hosts with lines exchange short flows, light gray end-hosts with lines exchange long flows that are policed by links \(l_5\), \(l_{14}\), and \(l_{20}\) (this is the class-\(c_2\) traffic), and white end-hosts exchange a mix of short and long flows, but they do not participate in the measurements (they provide background traffic). Table 3 shows the exact parameters for each type.

**Results.**

In this experiment, our algorithm suffers no false-positives and no false-negatives (each of the three policing links appears in at least one link sequence that is classified as non-neutral). It achieves granularity 2.7.

Figure 10(a) summarizes the actual performance of each link with respect to each path. These numbers are ground truth, directly measured by the network; our algorithm does not use them in any way. For each link, we show two boxplots: the left one summarizes the link’s actual performance for class \(c_1\), and the right one summarizes its actual performance for class \(c_2\). To create each boxplot, we measure the performance of the link with respect to every individual path traversing the link. For instance, consider link \(l_{20}\): according to the figure, this link never introduces congestion into class-\(c_1\) paths (left boxplot), while it introduces congestion on average 4% of the time into class-\(c_2\) paths (right boxplot).

We see that, for links \(l_5\), \(l_{14}\), and \(l_{20}\), the two boxplots are significantly further apart than for the rest of the links. This validates the basic premise of our model: links implementing traffic differentiation have significantly higher performance-number variability than the rest of the links.

Figure 10(b) summarizes the performance of different link sequences as inferred by our algorithm. In this network,
Figure 9: Experiment topology B. Dark gray end-hosts with lines exchange short flows. Light gray end-hosts with lines exchange long flows, which are policed by links $l_5$, $l_{14}$, and $l_{20}$. White end-hosts exchange a mix of short and long flows, but do not participate in the measurements.

Figure 10: Ground truth and algorithm results for topology B. Parameters in Table 3.

6.5 Conclusions

Our algorithm did not misclassify a neutral link sequence...
as non-neutral in any of our experiments, even when half the paths traversing that link carried 1Mb flows and the other half carried 10Gb flows. This is because our performance metric is robust to TCP dynamics: The performance number of a path indicates how often it suffers non-negligible packet loss, not how much packet loss it suffers. TCP dynamics may cause the same link to introduce different amounts of packet loss during the same time interval; however, when the link does not explicitly differentiate between the two paths, it is unlikely to introduce non-negligible packet loss in one path and not in the other during the same time interval.

Congestion did not interfere with neutrality inference in any of our experiments. In the experiment on topology B, there exist both neutral and non-neutral, severely congested links. For instance, both links $l_{13}$ and $l_{14}$ operate close to capacity; if we look at packet loss and queue occupancy for these two links over time (Figure 11), there is no clue that $l_{14}$ applies traffic differentiation while $l_{13}$ does not. This does not affect the algorithm, because congestion on its own does not lead to significantly inconsistent external observations; only congestion that is preferentially inflicted on some paths does.

![Figure 11: Queue occupancy of a neutral and a non-neutral link from topology B.](image)

A key difference from network tomography is that we neither target nor require accurate inference of the performance of link sequences. For example, link $l_{20}$ has actual congestion probabilities 0 and 4% for the two classes (Figure 10(a)), whereas our algorithm infers that it has congestion probabilities 0 and 2% (Figure 10(b), link sequence #23). This is because the algorithm infers the link’s congestion probabilities for normalized traffic aggregates (Section 6.2), which are conservative estimates of the link’s congestion probabilities for the two performance classes. Despite this, our algorithm correctly classifies the link as non-neutral, because it uses clustering: a link sequence $\lambda$ is classified as non-neutral as long as its inferred performance numbers (in this case 0 and 2%) are sufficiently different that $\lambda$ is assigned to the high-unsolvability cluster (Section 6.2).

As a side-note, we found that modern TCP variants are designed to converge with minimal packet loss. This is detrimental to tomography algorithms that rely on packet-loss measurements, e.g., to identify bottleneck links. Our performance metric is robust to such cautious congestion control, because—as mentioned above—it is a function of the frequency, not severity of loss events. Cautious congestion control may enable all classes to converge with minimal packet loss; however, when a link implements traffic differentiation, it necessarily introduces more loss events into deprioritized traffic than the rest, even if each of these events is not severe.

We close by noting that we repeated our experiments with all the loss thresholds and measurement intervals stated in Table 1, and there was no significant change in the results.

7. **DISCUSSION**

In this section, we discuss open issues.

**Measurement platform.**

We assume the existence of a measurement platform and knowledge of the network graph that interconnects the measurement points. The most realistic deployment option is to use an existing platform where coalitions of end-hosts periodically measure the performance of the paths between them and upload the results to a centralized location for processing [4, 3, 24].

To discover the network topology, we can leverage existing work like Rocketfuel [27], AS-level traceroute [20], and manually collected ISP topologies [5]. Many of these proposals rely on IP traceroute, and one may ask: if we can use traceroute for discovering topology, why not use it also for measuring link or ISP performance (Section 1)? One reason is that a link can treat traceroute probes differently from other traffic (purposefully or not). That said, when we do deploy our algorithm, it will have to compare it against probe-based algorithms [19, 31].

Another challenge is collecting (performance and topology) measurements from sufficiently diverse vantage points: Today, the end-hosts that participate in measurement platforms are typically located in University or residential networks. By measuring the performance of the one-way paths between these end-hosts, one can detect neutrality violations against P2P traffic [15, 11] or against particular end-hosts. However, if we want to detect neutrality violations against a content provider, we also need to know the performance and topology of the one-way paths from this provider to various end-hosts (which we cannot measure directly, assuming the provider does not participate in the measurement platform). One option we are exploring is to infer the performance of one-way paths from round-trip performance based on TCP semantics [25], and to infer the topology of the one-way paths using reverse traceroute [17].

**Link independence.**

We assume that the status of each link is independent from the status of any other link. This simplifies our analysis and algorithm, because it allows us to express joint probabilities
as products of marginal probabilities (thus create our simple linear equations). We can relax this assumption at least in the following way: we can assume that the status of each link is independent from the status of any other link that belongs to a different administrative domain. This will change how we create System 3 and network slices, but there exists work (on applying tomography to correlated links [14]) that we can leverage to do so. Lemmas 1 and 2 will remain the same. Theorem 1 will change to refer to distinguishable sets of links (as opposed to distinguishable links); the proof will rely on the theory developed in [14] (as opposed to [22]).

**Correlated performance classes.**

Theorem 1 and Lemma 3 assume that: within any given time interval, if a non-neutral link introduces non-negligible packet loss in its top-priority class, it introduces non-negligible packet loss in the other classes as well. This assumption simplifies the definition of the equivalent neutral network, but it does not hold when a non-neutral link uses separate queues for different performance classes. Removing it will not affect our algorithm, but it will require certain theory changes. For instance, in the definition of the equivalent neutral network, we will have to distinguish two types of non-neutral links: (a) Those where performance numbers for different classes are correlated as in the assumption; a link that polices lower-priority classes would fall into this category. (b) Those where performance numbers for different classes are not correlated; a link that uses separate queues for different performance classes would fall into this category. Each non-neutral link of type (a) will be mapped to a set of virtual links as described in Section 3.2. Each non-neutral link of type (b) will be mapped to a set of parallel virtual links (to model the fact that the performance numbers for different classes are not correlated).

**Path versus flow differentiation.**

We define a non-neutral link as one that differentiates between traffic from different paths, as opposed to one that differentiates by traffic type (which is the typical definition). What happens when the network does differentiate by traffic type, e.g., throttles BitTorrent traffic?

In a realistic scenario, we expect our algorithm to detect differentiation by traffic type without requiring any changes. Consider a scenario where each path carries a mix of traffic types, but different paths carry different mixes. For instance, any path from a content provider to an end-host carries only non-BitTorrent traffic, whereas any path between two end-hosts carries both BitTorrent and non-BitTorrent traffic at different points in time. In this case, any link that differentiates against BitTorrent traffic also differentiates against paths that carry BitTorrent traffic (e.g., it drops packets from the these paths more often than from the rest); hence, differentiation by type results into differentiation by path, which is what our algorithm detects.

In the worst-case scenario, each path carries roughly the same mix of traffic types all the time. To deal with this case, we need to redefine a non-neutral link as one that differentiates between different traffic aggregates that follow the same path. This change of definition affects only the way we collect our external observations: in System 3, each element of the vector $\tilde{y}$ corresponds to a different set of traffic aggregates (as opposed to a different pathset). We can create traffic aggregates as follows: for each path, we measure the end-to-end performance experienced by each UDP or TCP flow, and we classify flows that experience the same performance in every time interval into the same traffic aggregate.

**Performance metrics.**

We restrict ourselves to additive performance metrics that can be defined not only for paths, but also for pathsets. What happens when the network violates neutrality, but that results in inconsistent latency or jitter—in general, performance metrics that cannot be defined for pathsets?

A promising approach is to convert our desired performance metric into one that can be defined for pathsets. For instance, if we want to detect neutrality violations that result in inconsistent latency, we can define a link’s performance as the probability that the link introduces latency below some pre-configured threshold; then we can define a pathset’s performance as the probability that all the links traversed by the pathset introduce latency below some pre-configured threshold.

Defining performance as a probability of a congestion event means that our external observations are measurements of the frequency with which paths and pathsets are congested. The limitation of our approach is that we cannot detect a neutrality violation if it lasts for one (in practice, a small number of) measurement interval(s). However, our detection is not probabilistic: as long as a neutrality violation lasts for a significant number of intervals (and it is externally observable), our approach can detect it.

**8. RELATED WORK**

Several proposals detect traffic differentiation based on transport-layer headers or payload [18, 31, 15, 11]. The common theme of these proposals is to have two end-hosts exchange two different traffic flows (e.g., a BitTorrent flow and a flow that contains random bytes) over the same network path; if the treatment of the two flows is significantly different, then the network path between end-user and monitoring server must be non-neutral. Unlike our algorithms, these systems were not designed to detect traffic differentiation that affects all flows of an end-host, nor localize it to specific links.

NetPolice [31] belongs to the above body of work, but moreover detects whether an ISP treats traffic differently based on routing information, e.g., previous- or next-hop AS. The main idea is to use traceroute-like probes to measure the loss rate inflicted by an ISP on traffic associated with different neighboring ASes; if these loss rates are significantly different from each other, then the ISP must be non-neutral. We focus, instead, on the scenario where we cannot rely on
traceroute-like probes (or any other mechanism) to directly measure the loss rates of links or link sequences (e.g., because there is no practical way of generating such probes, or probes are not treated the same as other traffic).

Nano [28] “detects whether an ISP causes performance degradation for a service when compared to performance for the same service through other ISPs.” We view this work and ours as complementary: it detects whether two different ISPs inflict different performance on the same kind of traffic; we detect whether any particular link (or link sequence) inflicts different performance on different traffic.

ShaperProbe [16] and Packsen [30] detect whether a network path is shaping an end-user’s traffic and also determine the parameters of the shaper. The main idea is to have the end-user send traffic to a monitoring server, while keeping track of the rate at which the server receives the user’s traffic; the evolution of the receiving rate can reveal the presence of a shaper between end-user and server, as well as the shaper’s properties. We view this work and ours as complementary: it detects whether any single flow is subjected to shaping and identifies the parameters of the shaper; we detect whether different traffic flows are subjected to different treatment (of any kind) and localize this differentiation to specific links.

9. CONCLUSION

We studied the problem of detecting network-neutrality violations and localizing them to specific links. We presented conditions under which neutrality violations are observable and non-neutral links are identifiable based solely on external observations. Based on our analysis, we proposed an algorithm that takes as input a network graph and end-to-end measurements, and it identifies non-neutral link sequences; we evaluated it using network emulation. Our results indicate that it is indeed possible to reason about network neutrality, even when we know nothing about the internal behavior of the network. We hope that this work is a small step toward making the Internet more transparent.

Acknowledgments. We would like to thank Constantine Dovrolis for encouraging us to pursue this work despite early difficulties; Udi Weinsberg, our shepherd Aditya Akella, and the 6 anonymous SIGCOMM reviewers, for their deep, thorough reviews; and Mihai Dobrescu, Jonas Fietz, Dimitri Melissa- yas, Pavlos Nikolopoulos, Patrick Thiran, for helping us improve the paper. This work was supported by the Swiss National Science Foundation (SNSF).

10. REFERENCES

APPENDIX A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G = (V, L)$</td>
<td>The network graph, with nodes $V$ and links $L$.</td>
</tr>
<tr>
<td>$L_n$</td>
<td>The set of all neutral links in $L$.</td>
</tr>
<tr>
<td>$L_R$</td>
<td>The set of all non-neutral links in $L$.</td>
</tr>
<tr>
<td>$P$</td>
<td>The set of all paths.</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>An arbitrary set of pathsets.</td>
</tr>
<tr>
<td>$C$</td>
<td>The set of all performance classes.</td>
</tr>
<tr>
<td>$l_k$</td>
<td>The $k$-th link, assuming an arbitrary ordering of the $</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>An arbitrary loop-free sequence of links in $L$.</td>
</tr>
<tr>
<td>$p$</td>
<td>An arbitrary path in $P$.</td>
</tr>
<tr>
<td>$\pi_i$</td>
<td>An arbitrary pathset (set of paths) in $P$.</td>
</tr>
<tr>
<td>$c_n$</td>
<td>The $n$-th class, assuming an arbitrary ordering of the $</td>
</tr>
<tr>
<td>$\text{Paths}(l)$</td>
<td>The set of paths that traverse link $l$.</td>
</tr>
<tr>
<td>$\text{Links}(p)$</td>
<td>The set of links traversed by path $p$.</td>
</tr>
<tr>
<td>$\text{Links}(\pi)$</td>
<td>The set of links traversed by at least one path from pathset $\pi$.</td>
</tr>
<tr>
<td>$x$</td>
<td>The performance number of an arbitrary link.</td>
</tr>
<tr>
<td>$x_k$</td>
<td>The performance number of neutral link $l_k$.</td>
</tr>
<tr>
<td>$\vec{x}$</td>
<td>The vector { $x_k \mid k = 1\ldots</td>
</tr>
<tr>
<td>$x(n)$</td>
<td>The performance number of an arbitrary non-neutral link for performance class $c_n$.</td>
</tr>
<tr>
<td>$\bar{x}(n)$</td>
<td>The performance number of non-neutral link $l_k$ for performance class $c_n$.</td>
</tr>
<tr>
<td>$y$</td>
<td>The performance number of an arbitrary pathset.</td>
</tr>
<tr>
<td>$y_i$</td>
<td>The performance number of pathset $\pi_i$.</td>
</tr>
<tr>
<td>$\vec{y}$</td>
<td>The vector { $y_i \mid i = 1\ldots</td>
</tr>
<tr>
<td>$\mathbf{A}(\Pi)$</td>
<td>The generalized routing matrix that consists of equations for all the pathsets in $\Pi$.</td>
</tr>
<tr>
<td>$\mathbb{P}(l)$</td>
<td>The probability that neutral link $l$ introduces negligible packet loss during any given time interval.</td>
</tr>
<tr>
<td>$\mathbb{P}(l, c_n)$</td>
<td>The probability that non-neutral link $l$ introduces negligible packet loss into $c_n$-traffic during any given time interval.</td>
</tr>
<tr>
<td>$\mathbb{P}(\lambda, c_n)$</td>
<td>The probability that link sequence $\lambda$ introduces negligible packet loss into $c_n$-traffic during any given time interval.</td>
</tr>
<tr>
<td>$\mathbb{P}(\pi)$</td>
<td>The probability that all paths in $\pi$ experience negligible packet loss during any given time interval.</td>
</tr>
</tbody>
</table>

Table 4: Symbols defined in Section 2.3.

Algorithm 2 Computation of pathset performance numbers

Input: $M[t][p]$, the set of packets sent in interval $t$ by path $p$.

Output: $y_i$, the performance number of pathset $\pi_i \in \Pi$.

1: for each interval $t = 1..T$ do
2: $m \leftarrow \infty$
3: for each path $p \in \text{Paths}(\lambda)$ do
4: if $\{M[t][p]\} < m$ then
5: $m \leftarrow \{M[t][p]\}$
6: end if
7: end for
8: for each path $p \in \text{Paths}(\lambda)$ do
9: $M[t][p] \leftarrow \{ m \text{ random packets from } M[t][p] \}$
10: $L[t][p] \leftarrow L[t][p] \cap M[t][p]$
11: if $\{L[t][p]\} < 0.01 \cdot m$ then
12: $S[t]\{(p)\} \leftarrow 1$
13: else
14: $S[t]\{(p)\} \leftarrow 0$
15: end if
16: end for
17: for each path pair $\{p_i, p_j\} \in \Pi$ do
18: $S[t]\{(p_i, p_j)\} \leftarrow S[t]\{(p_i)\} \wedge S[t]\{(p_j)\}$
19: end for
20: end for
21: for each pathset $\pi_i \in \Pi$ do
22: $y_i \leftarrow \sum_j S[t]\{\pi_i\}$
23: end for
24: return $\{y_i \mid \pi_i \in \Pi\}$

APPENDIX B

Proof of Theorem 1

The following result will be useful to prove Theorem 1.

**Lemma 4.** Given a network $G$ with $L$ and $P$, if all links in $L$ are distinguishable from each other, then $A(P^*)$ has full column rank.

**Proof.** In order to show $A(P^*)$ has full column rank, all we need to prove is: \( \vec{x} = \vec{x} \), if

\[ A(P^*) \cdot \vec{x} = A(P^*) \cdot \vec{x}, \tag{5} \]

then $\vec{x} = \vec{x}$. Clearly, the result is independent on the specific type of performance metrics, so we prove it based on the probability of congestion-free defined in Section 2.3. (i.e., $\vec{x} = \{ x_k \mid k = 1\ldots|L| \}$ and $x_k \equiv \log(\mathbb{P}(l_k))$, where $\mathbb{P}(l_k)$ is the probability that $l_k$ is congestion free.)

We use a boolean random variable $Y_i$ to represent the status of $p_i \in P$. If $p_i$ is congested, $Y_i = 1$; otherwise, $Y_i = 0$. For any $\pi \in P^*$, let $\bar{Y}(\pi) = \{ Y_i \mid p_i \in \pi \}$ denote the status of all the paths in $\pi$.

Given a specific path state vector $\vec{y} \in \{0, 1\}^{|P|}$, denote the corresponding set of good and bad paths by $P_G$ and $P_B$ respectively. Let $|\vec{y}| = |P_G|$ denote the Hamming weight of $\vec{y}$. Given $\vec{x}$, the probability that the path state is $\vec{y}$ is denoted
According to the relationship between the joint probability column rank.\[\vec{x} = \vec{x}'\]

We first prove, if (5) holds, then we have \[\mathbb{P}_x(\vec{y}(P) = \vec{y}) = \mathbb{P}_{x'}(\vec{y}(P) = \vec{y})\] for any \(\vec{y} \in \{0, 1\}^{|P| \times 1}\), by induction on the Hamming weight of \(\vec{y}\).

**Step 1:** When \(|\vec{y}| = 0\), i.e., \(\vec{y} = 0\), we have
\[
\mathbb{P}_x(\vec{y}(P) = 0) = \prod_{k=1}^{|L|} (1 - \mathbb{P}(l_k)) = \exp \left( \sum_{k=1}^{|L|} \log(1 - \mathbb{P}(l_k)) \right) = \exp(A(\{P\}) \cdot \vec{x}),
\]
where \(A(\{P\}) = 1\) is a row of \(A(P^*)\). Therefore, \(\forall \vec{x}\) and \(\vec{x}'\), equation (5) implies
\[
\mathbb{P}_x(\vec{y}(P) = 0) = \mathbb{P}_{x'}(\vec{y}(P) = 0).
\]

**Step 2:** We prove the conclusion for any \(\vec{y}\) such that \(1 \leq |\vec{y}| \leq |P| - 1\). Suppose that for any \(\vec{y}\) such that \(0 \leq |\vec{y}| < |\vec{y}|\), equation (5) implies \(\mathbb{P}_x(\vec{y}(P) = \vec{y}) = \mathbb{P}_{x'}(\vec{y}(P) = \vec{y})\). According to the relationship between the joint probability and the marginal probability, we have
\[
\mathbb{P}_x(\vec{y}(P) = \vec{y}) = \mathbb{P}_x(\vec{y}(P_G) = 0, \vec{y}(P_B) = 1) = \mathbb{P}_x(\vec{y}(P_G) = 0) - \sum_{0 < |\vec{y}| \leq 0} \mathbb{P}_x(\vec{y}(P_G \cup \pi) = 0, \vec{y}(P_B \setminus \pi) = 1).
\]

On the right hand side of (7), \(\mathbb{P}_x(\vec{y}(P_G) = 0) = \exp(A(\{P_G\}) \cdot \vec{x})\). \(A(\{P_G\})\) is a row of \(A(P^*)\), so equation (5) implies
\[
\mathbb{P}_x(\vec{y}(P_G) = 0) = \mathbb{P}_{x'}(\vec{y}(P_G) = 0),
\]
\[
\mathbb{P}_x(\vec{y}(P_G \cup \pi) = 0, \vec{y}(P_B \setminus \pi) = 1) \text{ can be expressed as} \mathbb{P}_x(\vec{y}(P_G \cup \pi) = \vec{y}), \text{ where } \vec{y}(P_G \cup \pi) = 0, \vec{y}(P_B \setminus \pi) = 1. \text{ As } \emptyset \cap \pi \subseteq P_G, \text{ then } |\vec{y}| = |P_G \cup \pi| = |P_G| - |\pi| < |\vec{y}|. \text{ According to the induction assumption, equation (5) implies}
\]
\[
\mathbb{P}_x(\vec{y}(P_G \cup \pi) = 0, \vec{y}(P_B \setminus \pi) = 1) = \mathbb{P}_{x'}(\vec{y}(P_G \cup \pi) = 0, \vec{y}(P_B \setminus \pi) = 1),
\]
Combining (7), (8) and (9), we have the conclusion when \(1 \leq |\vec{y}| \leq |P| - 1\).

**Step 3:** When \(|\vec{y}| = |P|\), i.e., \(\vec{y} = 1\), \(\mathbb{P}_x(\vec{y} = \vec{y}) = 1 - \sum_{\vec{y}' < |P|} \mathbb{P}_x(\vec{y}' = \vec{y}')\). According to step 1 and 2, (6) holds for \(|\vec{y}| = |P|\).

In [22], the authors have proved that “\(\mathbb{P}_x(\vec{y} = \vec{y}) = \mathbb{P}_{x'}(\vec{y} = \vec{y})\) for any snapshot \(\vec{y}\) always implies that \(\vec{x} = \vec{x}'\). Therefore, \(\forall \vec{x}\) and \(\vec{x}'\), equation (5) always implies that \(\vec{x} = \vec{x}\), i.e., \(\vec{x} = \vec{x}\). So we conclude that \(A(P^*)\) has full column rank.

We can now prove Theorem 1.

**Proof.** Without loss of generality, we suppose all links in \(L^+\) and \(L\) are distinguishable from each other. Otherwise, we merge each group of indistinguishable links to a single logical link. The performance number of the logical link is the sum of its member links.

Let \(\vec{x}^+\) denote the ground-truth performance numbers of the links in \(L^+\). \(\vec{x}^+\) is a column vector of length \(|L^+|\). The \(k\)-th element of \(\vec{x}^+\) corresponds to the \(k\)-th column of \(A^+\). As \(A^+\) correctly captures the relationships between \(\vec{x}^+\) and the external observations \(\vec{y}, \vec{y} = A^+(\Pi) \cdot \vec{x}^+, \forall \Pi \subseteq P^*\). Therefore, System 3 can be expressed as
\[
A^+(\Pi) \cdot \vec{x}^+ = A(\Pi) \cdot \vec{x},
\]
where \(\vec{x}\) are unknown variables.

We first prove that the condition is sufficient:

On the one hand, \(A^+(\Pi)\) includes all the columns of \(A(\Pi)\): the column of \(A(\Pi)\) corresponding to neutral (resp. non-neutral) link \(l\) is the same with the column of \(A^+(\Pi)\) corresponding to \(l^+\) (resp. \(l\)). On the other hand, when there exists at least one virtual link \(l^+(n) \in L^+\) that is distinguishable from any links in \(L\) \((n\) must not equal \(n^*\)), the column of \(A^+(\Pi)\) corresponding to \(l^+(n)\), denoted by \(a^+(n)\), is not in \(A(\Pi)\). Let \(\Pi = P^*\). According to Lemma 4, \(A^+(P^*)\) and \(A(P^*)\) have full column rank, so \(a^+(n)\) is not in the column space of \(A(P^*)\).

As \(n \neq n^*\), the performance number of \(l^+(n)\), \(x(n) - x(n^*) \neq 0\). Hence, \(A^+(P^*) \cdot \vec{x}^+\) is not in the column space of \(A(P^*)\), i.e., (10) does not have solution when \(\Pi = P^*\).

Therefore, there always exists a set of pathsets \(\Pi = P^*\), such that when there exists a virtual link \(l^+(n)\) satisfying the condition, System 3 is unsolvable, i.e., the network’s neutrality violation is observable.

Second, we show that the condition is necessary by contradiction:

If the condition does not hold, i.e., any virtual link in \(L^+\) are indistinguishable from some link in \(L\), then \(A^+(\Pi) = A(\Pi)\) for any \(\Pi \subseteq P^*\). In this case, it is obvious that (10) would be solvable for any \(\Pi \subseteq P^*\), and thus System 3 is always solvable.

Therefore, the network’s neutrality violation is unobservable when the condition does not hold.

**Construct System 4 for \(\lambda\)**

Here is how to construct System 4 for \(\lambda\).

1. We form a special set of pathsets \(\Pi_\lambda\) as follows:
   (a) We identify all path pairs \(p_i, p_j \in P\) such that \(\text{Links}(p_i) \cap \text{Links}(p_j) = \lambda\).
   (b) For each such path pair \(p_i, p_j\), we add \(\{p_i\}, \{p_j\}, \{p_i, p_j\}\) to \(\Pi_\lambda\).
2. Let \(\Pi_\lambda = \cup_{\pi \in \Pi_\lambda} \pi\). We map the sub-graph of \(G\), which consists of all the paths in \(\Pi_\lambda\), to a network slice \(G_\lambda = (\Lambda_\lambda, \mathcal{L}_\lambda, \pi_\lambda)\) as follows:
   (a) Link sequence \(\lambda\) is mapped to a single link in \(\mathcal{L}_\lambda\).
(b) \( \forall p_i \in \mathcal{P}_\lambda, \bar{x}_i = \text{Links}(p_i) \setminus \lambda \), which is the set of all links in \( p_i \) outside \( \lambda \), is mapped to a single link in \( \mathcal{L}_\lambda \).

(c) Any path \( p_i \in \mathcal{P}_\lambda \) is still an end-to-end path in \( \mathcal{G}_\lambda \). In fact, \( \mathcal{G}_\lambda \) is a logical tree with two levels. The logical link corresponding to \( \lambda \) is in the first level and others are in the second level. In \( \mathcal{G}_\lambda \), any \( p_i \in \mathcal{P}_\lambda \) consists of two logical links corresponding to \( \lambda \) and \( \bar{\lambda}_i \), respectively.

3. We form a system of equations \( \bar{y} = A_\lambda(\Pi_\lambda) \cdot \bar{x} \), where

(a) \( \bar{y} = \{y_i| i = 1..|\Pi_\lambda|\} \) is the vector of performance numbers of the paths set in \( \Pi_\lambda \), which is the external observations.

(b) \( \bar{x} = \{x_{\lambda'}| \lambda' = \lambda \text{ or } \lambda' = \bar{\lambda}_i (\forall p_i \in \mathcal{P}_\lambda)\} \) is the vector of the performance number of the logical links in \( \mathcal{G}_\lambda \).

(c) \( A_\lambda(\Pi_\lambda) \) is the routing matrix of \( \Pi_\lambda \) in \( \mathcal{G}_\lambda \).

**Proof of Lemma 2**

**Proof.** We prove the conclusion by showing that if \( \lambda \) is neutral, System 4 is solvable.

Suppose \( \lambda \) is neutral, we construct a special \( \bar{x} = \bar{x}_0 \) as follows, where \( x_{\lambda} \) (or \( x_{\bar{\lambda}}(n) \)) are the ground-truth performance numbers of link \( l_k \):

1. Let \( x_{\lambda} = \sum_{l_k \in \lambda} x_k \).

2. \( \forall p_i \in \mathcal{P}_\lambda \cap c_n, n = 1..|\mathcal{L}| \), let \( x_{\lambda} = \sum_{l_k \in \lambda \cap L_n} x_k + \sum_{l_k \in \lambda \cap L_n} x_{k}(n) \).

We show that \( \bar{x}_0 \) is always one solution of System 4. Actually, for any \( \pi_i \in \Pi_\lambda \):

1. If \( \pi_i = \{p_i\} \), the corresponding equation in (4) is
   \[ y_{i} = x_{\lambda} + x_{\lambda_j}, \tag{11} \]
   \( y_i \) is the external observation of \( \pi_i \) in the original network \( G \), suppose \( p_j \in c_n \), then the left hand side of (11) is
   \[ \sum_{l_k \in \lambda} x_k + \sum_{l_k \in \lambda \cap L_n} x_k + \sum_{l_k \in \lambda \cap L_n} x_{k}(n). \]
   Clearly, \( \bar{x}_0 \) is a solution of (11).

2. If \( \pi_i = \{p_{j1}, p_{j2}\} \), the corresponding equation in (4) is
   \[ y_{i} = x_{\lambda} + x_{\lambda_{j1}} + x_{\lambda_{j2}}. \tag{12} \]

Suppose \( p_{j1} \in c_n, p_{j2} \in c_m, \) as \( p_{j1} \cap p_{j2} = \lambda \), the left hand side of (12) is
   \[ \sum_{l_k \in \lambda} x_k + \sum_{l_k \in \lambda_{j1} \cap L_n} x_k + \sum_{l_k \in \lambda_{j1} \cap L_n} x_{k}(n) + \sum_{l_k \in \lambda_{j2} \cap L_n} x_k + \sum_{l_k \in \lambda_{j2} \cap L_n} x_{k}(m). \]

We can verify that \( \bar{x}_0 \) is also a solution of (12).

It is therefore that \( \bar{x}_0 \) is a solution of every equation of System 4, and hence it is a solution of System 4.

When \( \lambda \) is neutral, System 4 has a solution \( \bar{x}_0 \). In other words, if System 4 is unsolvable, \( \lambda \) is non-neutral.

**Proof of Lemma 3**

**Proof.** For any path pair \( \pi = \{p_i, p_j\} \in \Pi_\lambda \), we obtain a sub-system of System 4 that consists of the equations corresponding to \{ \( p_i \) \} \{ \( p_j \) \} and \( \pi \):

\[
\begin{align*}
\{p_i\} : & \quad y_i = x_{\lambda} + x_{\lambda_k} \\
\{p_j\} : & \quad y_j = x_{\lambda} + x_{\lambda_k} \\
\pi : & \quad y_{\pi} = x_{\lambda} + x_{\lambda_k} + x_{\lambda_\pi}.
\end{align*}
\]

The unique solution of \( x_{\lambda} \) in (13) is
\[
x_{\lambda} = y_i + y_j - y_{\pi}. \tag{14}
\]

Let \( \hat{x}_{\lambda}(n) \) denote the ground-truth performance number of \( \lambda \) for \( c_n \in C, \forall \lambda' \subset L, n = 1..|\mathcal{L}| \). Then
\[
\hat{x}_{\lambda}(n) = \sum_{l_k \in \lambda \cap L_n} x_k + \sum_{l_k \in \lambda \cap L_n} x_{k}(n). \tag{15}
\]

If \( \lambda' \cap L_n = \emptyset \), then \( \hat{x}_{\lambda}(n) = \sum l_k \in \lambda \cap L_n x_{k}(n) \) for any \( n = 1..|\mathcal{L}| \). For any path \( p \in c_n \), its performance number is
\[
y = \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(n). \tag{16}
\]

When \( \pi \subset c_n \), i.e., \( p_i \) and \( p_j \) are in the same class, we have
\[
y = \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(n) + \hat{x}_{\lambda_j}(n). \tag{17}
\]

Substituting (16)(\( p = p_i, p_j \)) and (17) to (14), we have
\[
x_{\lambda} = \hat{x}_{\lambda}(n). \tag{18}
\]

When \( p_i \) and \( p_j \) are in different classes, without loss of generality, we suppose \( p_i \in c_n, p_j \in c_m, n \neq m \). Then, with the aid of equivalent neutral network, we have
\[
y_{\pi} = \sum_{l_k \in \lambda \cap L_n} x_k + \sum_{l_k \in \lambda \cap L_n} x_{k}(n) + \sum_{l_k \in \lambda \cap L_n} x_{k}(n) + \sum_{l_k \in \lambda \cap L_n} x_{k}(n)
\]
\[
\quad + \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(m) = \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(m) + \hat{x}_{\lambda}(n) + \hat{x}_{\lambda}(m) \tag{19}
\]

Substituting (16)(\( p = p_i, p_j \)) and (19) to (14), we have
\[
x_{\lambda} = \hat{x}_{\lambda}(n). \tag{20}
\]

When the condition is satisfied, \( \pi_j \not\subset c_n \) implies that either (a): \( \pi_j \subset c_m, m \neq n \) or (b): the paths in \( \pi_j \) are not in the same class. In case (a), according to (18), the estimates of \( x_{\lambda} \) obtained from \( \pi_i \) and \( \pi_j \) are \( \hat{x}_{\lambda}(n) \) and \( \hat{x}_{\lambda}(m) \) respectively. In case (b), according to (18) and (20), the estimates of \( x_{\lambda} \) obtained from \( \pi_i \) and \( \pi_j \) are \( \hat{x}_{\lambda}(n) \) and \( \hat{x}_{\lambda}(n) \) respectively. As \( n \neq n^* \) and \( m \neq m^* \), two estimates of \( x_{\lambda} \) are always different.

Therefore, we get different solutions of \( x_{\lambda} \) from different sets of equations of System 4, each of which is unique in the
corresponding sub-system. System 4 is unsolvable and thus \( \lambda \) is identifiable. \( \square \)