

Principles of Safe Policy Routing Dynamics

Samuel Epstein, Karim Mattar, Ibrahim Matta
 Department of Computer Science
 Boston University
 {samepst, kmattar, matta}@cs.bu.edu

Technical Report No. BUCS-TR-2009-13
 This report revises Technical Report BUCS-2008-017.

Abstract—We introduce the *Dynamic Policy Routing (DPR)* model that captures the propagation of route updates under arbitrary changes in topology or path preferences. DPR introduces the notion of *causation chains* where the route flap at one node causes a flap at the next node along the chain.

Using DPR, we model the Gao-Rexford (economic) guidelines that guarantee the safety (*i.e.*, convergence) of policy routing. We establish three principles of safe policy routing dynamics. The *non-interference principle* provides insight into which ASes can directly induce route changes in one another. The *single cycle principle* and the *multi-tiered cycle principle* provide insight into how cycles of routing updates can manifest in any network.

We develop INTERFERENCEBEAT, a distributed algorithm that propagates a small token along causation chains to check adherence to these principles. To enhance the diagnosis power of INTERFERENCEBEAT, we model four violations of the Gao-Rexford guidelines (*e.g.*, transiting between peers) and characterize the resulting dynamics.

I. INTRODUCTION

The Border Gateway Protocol (BGP) is currently the de-facto inter-domain routing protocol employed in the Internet. BGP allows Autonomous Systems (ASes), operated by different administrative domains (*e.g.*, Internet Service Providers, companies, universities) to independently apply local policies for selecting routes and propagating routing information. Given the critical role and global scope of BGP, both its transient and steady-state performance have received significant attention, and problems related to delayed convergence [1] and potential instability [2], [3] (*i.e.*, route oscillations/flaps) have been identified and studied.

Route flaps in particular can be highly disruptive given the associated cost of communication and processing overheads. Route flaps can be transient (*i.e.*, short-term) due to temporary changes in topology or route/path preferences. Route flaps can also be persistent due to conflicting routing policies across ASes (*i.e.*, policies can not be simultaneously satisfied) [4].

Economic constraints that are typical of commercial relationships between ASes in the Internet—henceforth referred to as the Gao-Rexford guidelines [5]—have been shown to make BGP free from policy conflicts (*i.e.*, convergent). We refer to routing policy instances that adhere to the Gao-Rexford guidelines as *safe* and ones that do not as *potentially unsafe*. The Gao-Rexford guidelines are:

- 1) An AS classifies its neighboring ASes as either customer, peer or provider.

- 2) The path preferences are restricted in a hierarchical fashion. Every AS prefers a path through a customer AS over a path through a peer/provider AS.
- 3) All advertised paths are “valley-free”. They consist of zero or more customer-to-provider links followed by an optional peering link followed by zero or more provider-to-customer links.

Our Contribution:

We extend the Stable Paths Problem [4] (a static model of BGP) to capture the propagation dynamics of route updates under arbitrary changes in topology (*e.g.*, link failures) or path preferences (*e.g.*, policy configuration updates). We call this extended model the *Dynamic Policy Routing (DPR)* model. DPR introduces the notion of *causation chains* where the route flap at one node causes a flap at the next node along the chain.

We model a strict version of the Gao-Rexford guidelines which we call the *economic DPR* model. We prove the existence of several invariant properties of causation chains irrespective of arbitrary changes in topology or path preferences. For example, we prove that all causation chains in the economic DPR model are valley-free, thus generalizing the result in [6] to dynamic networks. Violations of the economic DPR model result in potentially unsafe routing behavior where the causation chains are not necessarily valley-free.

We develop INTERFERENCEBEAT, a distributed algorithm that checks if the routing dynamics adhere to the ones predicted by the economic DPR model. If not, then the presence of policy violations can be inferred. INTERFERENCEBEAT appends a token to each routing update message. Tokens are propagated along causation chains.

We model four common policy violations (*e.g.*, transiting between peers). For each violation, we prove the invariant properties of the resulting causation chains. Using these inferred properties, we extend the diagnosis power of INTERFERENCEBEAT. The novelty of this work is that:

- 1) We identify key principles (*i.e.*, invariant properties) of safe policy routing dynamics regardless of changes to the underlying topology or path preferences.
- 2) We identify and model four common violations of safe policy routing and characterize the resulting dynamics.
- 3) We introduce INTERFERENCEBEAT, a distributed algorithm to detect and diagnose policy violations.

II. PRINCIPLES OF SAFE POLICY ROUTING DYNAMICS

In this section, we distill the key results of our DPR model into three principles. These principles capture invariant properties of the routing dynamics under safe policy routing (*i.e.*, where the policies of all nodes adhere to the Gao-Rexford guidelines). We discuss reasons why ASes violate these guidelines. Such policy violations result in potentially unsafe routing dynamics where our principles no longer hold.

We also show that routing dynamics need to be explicitly considered when detecting policy violations. We postpone formal definitions to later sections and focus here on presenting the main intuitions behind our results.

A. What are the principles?

Non-Interference Principle: *If an AS y is not at a higher tier-level than (provider to) any two of its neighbors x and z , then x and z cannot directly induce path changes in each other through y . This principle holds regardless of changes in the underlying topology or path preferences.*

The notion of “inducing path changes” is synonymous with a continuous propagation of path changes across nodes, which we model in DPR as a causation chain. The basic premise of the non-interference principle comes from a result in DPR (Theorem 1 in Section IV) where we proved that any causation chain must not contain sequences such as a provider-to-customer-to-provider.

Figure 1 outlines all the Internet configurations where AS x cannot directly affect AS z through AS y . More specifically, non-interference holds if:

- 1) AS y is multi-homed with providers AS x and AS z .
- 2) AS y is a customer of AS x and a peer of AS z .
- 3) AS y is a peer of AS x and a customer of AS z .
- 4) AS y is a peer of both AS x and AS z

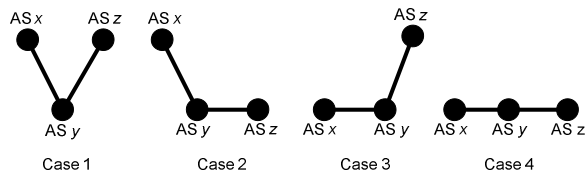


Fig. 1. All Internet configurations where AS x cannot directly affect AS z . Horizontal edges represent peering links and diagonal edges represent customer-to-provider links.

Single Cycle Principle: *In any cycle of routing update messages between ASes, every AS x affects its neighbor y at most once. This principle holds regardless of changes in the underlying topology or path preferences.*

The notion of “cycle” is synonymous with a continuous propagation of path changes across nodes where at least one node is affected twice. We model such a cycle of path changes in DPR as a *causation cycle*. The single cycle principle comes from a result in DPR (Theorem 2 in Section IV) where we proved that any causation cycle in safe policy routing occurs only once.

Multi-Tiered Cycle Principle: *Every cycle of routing update messages between ASes must have at least two ASes in different tier-levels. This principle holds regardless of changes in the underlying topology or path preferences.*

The multi-tiered cycle principle comes from a result in DPR (Theorem 2 in Section IV) where we proved that no causation cycle in safe policy routing can occur exclusively between peering ASes.

B. Why do the principles not always hold?

Violations of safe policy routing (*i.e.*, the Gao-Rexford guidelines) result in unpredictable, black-box dynamics that are potentially unsafe. When policy violations occur, the principles no longer hold (Table III in Section VI). The reasons for such violations are:

- 1) **Intentional:** representing legitimate policy configurations for backup links or complex agreements [7].
- 2) **Unintentional:** representing misconfigurations or complex real-time interactions between routers that do not reflect the intentions of the administrators.

C. How do we check the principles?

Network administrators can *statically* check whether they are conforming to the Gao-Rexford guidelines where the dynamics are guaranteed to conform to the principles. This can be done by inspecting their local preferences and ensuring that all adopted paths are valley-free.

Static checks are inadequate since not all nodes are necessarily compliant with the guidelines. Figure 2 illustrates “interference” between nodes 1 and 3. The interference is due to policy violations by node 2 which cannot be statically checked by node 3. Instead, node 3 will need to discover the interference by somehow detecting the causation chain propagating through nodes 1, 2 and 3.

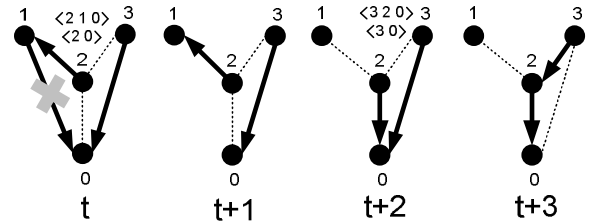


Fig. 2. Sample dynamics where interference occurs. The list of path preferences for nodes 2 and 3 are organized such that the most preferred path is at the top. Paths not explicitly listed are forbidden. All nodes are trying to reach destination node 0.

Node 3 is abiding by the Gao-Rexford guidelines and initially uses the customer path $\langle 30 \rangle$ which is valley-free. Node 2, however, violates the guidelines by preferring a path through its provider $\langle 210 \rangle$ over a path through its customer $\langle 20 \rangle$. At time t , the link connecting node 1 to node 0 is lost, causing node 1 to have an empty path to node 0 at time $t+1$. At time $t+2$, node 2 switches from path $\langle 210 \rangle$ to $\langle 20 \rangle$. This action in turn causes node 3 to switch from path $\langle 30 \rangle$ to $\langle 320 \rangle$ at

time $t + 3$. Even though node 3 abides by the Gao-Rexford guidelines, the forbidden interference occurs. The causation chain consists of a provider (node 1), followed by its customer (node 2), followed by another provider (node 3).

If node 2 does not violate the guidelines, the dynamics would manifest differently. Suppose the path $\langle 20 \rangle$ is forbidden, forcing node 2 to use its provider path $\langle 210 \rangle$. The loss of link connectivity between nodes 1 and 0 at time t causes node 2 to lose connectivity at time $t + 2$. Node 3 is unaffected. The causation chain solely consists of a provider (node 1) followed by its customer (node 2). Since this chain is valley-free, the dynamics conform to the principles.

III. DYNAMIC POLICY ROUTING MODEL

The Dynamic Policy Routing (DPR) model is used to capture the dynamics of BGP. Each AS is represented by a node in a graph. AS path preferences are represented by a ranking relation. DPR extends the notion of SPP [4] to model time-varying topologies and path preferences.

A. Basics of DPR

Definition 1 (Time). Time is represented by a non-negative, discrete index t such that: $t \in [0, \infty)$.

Definition 2 (Network). The network is represented by a graph $G = (V, E)$:

- Each vertex $u \in V$ represents an AS.
- Each edge in E is time dependent: $(u, v)^t \in E$ if u is connected to v at time t . Conversely, a lack of connectivity between u and v at time t (i.e., link failure) is represented by $(u, v)^t \notin E$.

There exists a distinguished destination node, represented as $root$, where $root \in V$.

Definition 3 (Paths). Paths are sequences of nodes of the form: $\langle u_1 u_2 \dots u_k \rangle$. The empty path is denoted by $\langle \rangle$. All paths end with the $root$ node. A concatenation of a node u with a path Q is represented as: $P = \langle u Q \rangle$. A path originating from u is represented by P^u . The set of paths originating from u is represented by \mathcal{P}^u .

Definition 4 (Path Preferences). At each time t , each node u has a unique preference over paths originating at u . This dynamic ranking is represented by the \succeq^t operator. If u prefers P^u over Q^u at time t then: $P^u \succeq^t Q^u$. If u prefers P^u over Q^u for all t then: $P^u \succeq Q^u$. Strict preference is defined by:

$$P^u \succ^t Q^u \text{ iff } P^u \succeq^t Q^u \text{ and } Q^u \not\succeq^t P^u$$

For all times t , for each node $u \in V$, \succeq^t is a total order over $\mathcal{P}^u \cup \langle \rangle$. Thus each node u has an ordered preference over all its paths to $root$. If two paths start with different nodes, then they have no preference relation. Forbidden paths P are those ranked below the empty path for all times: $\langle \rangle \succ P$. All paths with repeating nodes are forbidden.

Definition 5 (DPR Instance). A Dynamic Policy Routing (DPR) instance consists of a graph and a path preference $D = (\succeq^t, G)$.

Definition 6 (Best Paths). At each time index t , every node u has a path to $root$, represented by $P^u = \pi(u, t)$. The available path choices of a node, via all possible neighbors v , are represented by $\text{Choices}(u, t)$ where:

$$\text{Choices}(u, t) = \langle \rangle \cup \{ \langle u \pi(v, t) \rangle; (u, v)^t \in E \}$$

The $\text{Best}(u, t)$ notation represents the current best path for u :

$$\text{Best}(u, t) = \max_{\succeq^t} \text{Choices}(u, t)$$

The paths assigned to nodes at each time t is their best path of the previous round. For all nodes $u \in V$:

- $\pi(u, 0) = \langle \rangle$
- $\pi(u, t) = \text{Best}(u, t - 1)$

The path used by node u at time t , $\pi(u, t)$, was its best path at time $t - 1$, $\text{Best}(u, t - 1)$. This best path was determined using the ranking \succeq^{t-1} .

Definition 7 (Next-Hop Neighbor). The ρ notation is used to represent the next-hop neighbor of a current path:

$$\rho(u, t) = \text{NextHop}(\pi(u, t))$$

Definition 8 (Realized Paths). A path P^u is realized iff there exists a time t such that $\pi(u, t) = P^u$.

Proposition 1 (Path Deconstruction). If $\rho(u_0, t) = u_1$ then $\pi(u_0, t) = \langle u_0 \pi(u_1, t - 1) \rangle$

Proof: By the definition of π , $\pi(u_0, t) = \text{Best}(u_0, t - 1)$ so $\pi(u_0, t) \in \text{Choices}(u_0, t - 1)$. So by the definition of Choices , $\pi(u_0, t) = \langle u_0 \pi(u_1, t - 1) \rangle$, where $u_1 = \rho(u_0, t)$. ■

B. Causation in DPR

Definition 9 (Path Rank Changes). The following definitions describe the relative change in the rankings of selected paths for a node:

$$\begin{aligned} \text{RankDec}(u, t) & \text{ iff } \pi(u, t) \succ^t \pi(u, t + 1) \\ \text{RankInc}(u, t) & \text{ iff } \pi(u, t) \prec^t \pi(u, t + 1) \\ \text{RankSame}(u, t) & \text{ iff } \pi(u, t) = \pi(u, t + 1) \end{aligned}$$

The relative change in rankings are with respect to the current path ranking \succeq^t .

Definition 10 (Causation Function). In DPR, a node u may change its current path at a given time t . The causation function represents u 's neighboring node v responsible for u 's path change. Causation function is the base construct from which causation chains will be built. A causation function C maps each node u at a given time t to a neighboring node v : $C(u, t) = v$.

The operating conditions for the causation function are outlined in Table I. There are three cases for the causation function $C(u, t) = v$:

- 1) Node v was the next hop of u 's chosen path at time t . However, node v changed its path at time t , causing u to choose a less preferred path at time $t + 1$.
- 2) Node v advertised a new path at time t , causing u to choose a more preferred path through v at time $t + 1$.

TABLE I
CAUSATION FUNCTION

Condition 1: RankDec(u, t)	\Rightarrow	$C(u, t) = \rho(u, t)$
Condition 2: RankInc(u, t)	\Rightarrow	$C(u, t) = \rho(u, t + 1)$
Condition 3: RankSame(u, t)	\Rightarrow	$C(u, t)$ is empty

3) v is empty, because u 's path did not change between times t and $t + 1$.

Definition 11 (Causation Chain). A causation chain is a sequence of nodes where each node y_{i-1} causes y_i to change its current path. It is represented by $Y = \langle y_0 y_1 \dots y_k \rangle^t$, where:

$$C(y_i, t + i) = y_{i-1} \quad \text{for all } 0 < i \leq k$$

Time t is defined with respect to y_0 , and it takes i time steps to build the causation chain up to node y_i . An example of a causation chain can be seen in Figure 3.

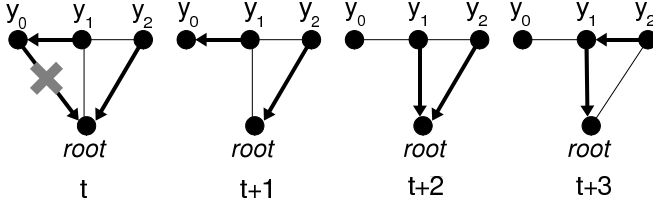


Fig. 3. Causation chain $Y = \langle y_0 y_1 y_2 \rangle^t$. A link failure between y_0 and $root$ occurred at time t , causing y_0 to have no path to $root$ at time $t + 1$. This causes y_1 to switch to a less preferred path at time $t + 2$, where $C(y_1, t + 1) = y_0$ with causation condition 1. This causes y_2 to switch to a more preferred path via y_1 at time $t + 3$, where $C(y_2, t + 2) = y_1$ with causation condition 2.

Definition 12 (Causation Cycle). A causation cycle is a causation chain with a repeated node: $Y = \langle y_0 y_1 \dots y_k \rangle^t$, where $y_0 = y_k$. The *primary* node of the causation cycle is $y_0 = y_k$.

Definition 13 (Simple Causation Cycle). A causation cycle Y is *simple* if:

$$C(y_1, t + k + 1) \neq y_0$$

The following examples represent simple and non-simple causation cycles:

$$\begin{aligned} \text{Simple:} & \quad \langle y_0 y_1 y_2 y_0 y_3 \rangle^t \\ \text{Non-Simple:} & \quad \langle y_0 y_1 y_2 y_0 y_1 \rangle^t \end{aligned}$$

IV. ECONOMIC DPR MODEL

This section will show that if a DPR instance conforms to a strict version of the Gao-Rexford guidelines [5], then its dynamic behavior can be characterized, regardless of changes in topology or path preferences. In particular, we show that all causation chains have the property known as ‘‘valley-free’’ and all causation cycles are simple. The economic constraints we consider are as follows:

1) Every node is customer, peer, or provider to its neighboring nodes.

- 2) A node cannot be a provider to itself. There are no customer-provider cycles. Furthermore, a node cannot be both a (direct or indirect) provider and a (direct or indirect) peer to another node.
- 3) For all times, each node prefers a path through a customer over a path through a peer/provider and prefers a path through a peer over a path through a provider.
- 4) Each node provides transit service only to its customers. Thus, all paths are valley-free.

These economic constraints are a stricter version of the Gao-Rexford guidelines which are sufficient to guarantee stability in a static graph. Thus, the economic DPR model is safe. The restrictions of the economic model enable equivalence classes of peers, as seen in Figure 4. The economic relationships between nodes can be represented using a pre-order relation.

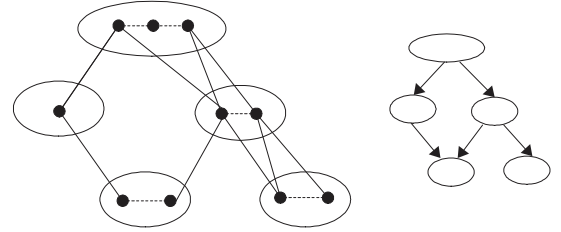


Fig. 4. Equivalence classes of peers in economic DPR.

A. Basics of Economic DPR

Definition 14 (Economic Operator). The economic relationship between nodes are described using the operator \succeq_{\S} . This operator is essential for reasoning about the economic relationships between nodes in both paths and causation chains. A *strict* economic relation is defined by:

$$u \succ_{\S} v \text{ iff } u \succeq_{\S} v \text{ and } u \not\preceq_{\S} v$$

and an equivalence relation is defined by:

$$u =_{\S} v \text{ iff } u \succeq_{\S} v \text{ and } u \preceq_{\S} v$$

Economic relationships can be derived from the operator \succeq_{\S} :

- If u is a customer of v then $u \prec_{\S} v$.
- If u is a provider to v then $u \succ_{\S} v$.
- If u is a peer to v then $u =_{\S} v$.

The properties of the economic operator \succeq_{\S} can be modelled using pre-order conditions:

- 1) (reflexive) $x \succeq_{\S} x$
- 2) (transitive) $x \succeq_{\S} y$ and $y \succeq_{\S} z$ implies $x \succeq_{\S} z$

The following transitive relationships hold:

$$x \succ_{\S} y \text{ and } y \succeq_{\S} z \text{ implies } x \succ_{\S} z$$

$$x \succeq_{\S} y \text{ and } y \succ_{\S} z \text{ implies } x \succ_{\S} z$$

Definition 15 (Customer, Peer, and Provider Paths). We define paths by the economic relationship between a path’s starting node u and its next-hop. For all paths P^u :

$$\begin{aligned} \text{Customer}(P^u) & \text{ iff } u \succ_{\S} \text{NextHop}(P^u) \\ \text{Peer}(P^u) & \text{ iff } u =_{\S} \text{NextHop}(P^u) \\ \text{Provider}(P^u) & \text{ iff } u \prec_{\S} \text{NextHop}(P^u) \end{aligned}$$

Definition 16 (Valley). We define a valley to be a sequence of three nodes $\langle a b c \rangle$ satisfying the condition:

$$a \succeq_{\S} b \preceq_{\S} c$$

The four types of valleys can be seen in Figure 5. Every valley-free sequence is a series of zero or more ascending customer-to-provider relationships, followed by an optional peer relationship, followed by a series of zero or more descending provider-to-customer relationships.



Fig. 5. Valleys

Definition 17 (Economic DPR Instances). An economic DPR instance $(\succeq_{\S}, \succeq^t, G)$ satisfies the following conditions:

- 1) All paths which have a valley are forbidden.

$$\text{HasValley}(P) \Rightarrow \langle \rangle \succ P$$

- 2) Customer paths are always preferred over peer/provider paths and peer paths are always preferred over provider paths. Thus given paths P_1^u and P_2^u :

$$\begin{aligned} \text{Customer}(P_1^u) \text{ and not Customer}(P_2^u) &\Rightarrow P_1^u \succ P_2^u \\ \text{Peer}(P_1^u) \text{ and Provider}(P_2^u) &\Rightarrow P_1^u \succ P_2^u \end{aligned}$$

B. Causation in Economic DPR

This section characterizes causation chains and cycles for economic DPR instances.

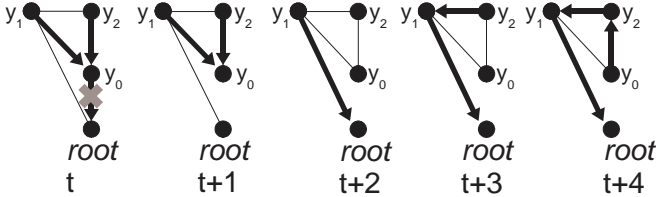


Fig. 6. Causation cycle $Y = \langle y_0 y_1 y_2 y_0 \rangle^t$. A link failure between y_0 and $root$ occurred at time t , causing y_0 to have no path to $root$ at time $t + 1$. This causes y_1 to switch to a less preferred path at time $t + 2$, where $C(y_1, t + 1) = y_0$ with causation condition 1. This causes y_2 to switch to a path through y_1 at time $t + 3$, where $C(y_2, t + 2) = y_1$ with causation condition 2. The cycle is closed with y_0 switching to a path via y_2 at time $t + 4$, where $C(y_0, t + 3) = y_2$ with causation condition 2. Note the existence of a separate causation chain $Y' = \langle y_0 y_2 \rangle^t$.

Theorem 1. Every causation chain of an economic DPR instance $(\succeq_{\S}, \succeq^t, G)$ is valley-free.

For ease of exposition, the full proof of Theorem 1 is in Appendix A. In the proof, we assume that there exists a causation chain that has a valley consisting of three consecutive nodes $\langle a b c \rangle^t$. First we prove that at no time during the causation chain did b have a customer path. Then we prove that at some time during the causation chain, c had a path through b . Since b is a customer/peer to c and b does not have a customer path then c had a realized valley path through b , causing a contradiction.

We introduce the following types of cycles:

Definition 18 (Horizontal Cycle). A causation cycle is horizontal if all adjacent nodes in the cycle are peers.

Definition 19 (Vertical Cycle). A causation cycle is vertical if there is at least one customer/provider relationship between adjacent nodes in the cycle.

Figure 6 represents a simple vertical causation cycle, where node y_0 loses a path to $root$ and reroutes through y_2 .

Lemma 1. Given a causation cycle $Y = \langle y_0 \dots y_k \rangle^t$ of an economic DPR instance $(\succeq_{\S}, \succeq^t, G)$, every node in Y is a provider to the primary node y_0 .

Proof: Let $y_i \in Y$, where $0 < i < k$. By Theorem 1, Y is valley-free and either $y_{i-1} \preceq_{\S} y_i$ or $y_i \succeq_{\S} y_{i+1}$. If the first case is true, then by the definition of valley-free paths $y_{j-1} \prec_{\S} y_j$ for all $0 < j < i$, and by the transitive nature of economic relationships, $y_0 \prec_{\S} y_i$. If the second case is true, then by the definition of valley-free paths $y_j \succ_{\S} y_{j+1}$ for all $i < j < k$, and by the transitive nature of economic relationships, $y_i \succ_{\S} y_k$. Thus every node y_i is a provider to $y_0 = y_k$. ■

Theorem 2. Every causation cycle $Y = \langle y_0 \dots y_k \rangle^t$ of an economic DPR instance is vertical and simple.

Proof: Lemma 1 directly implies that every causation cycle in economic DPR instances are vertical. The second part regarding simple causation cycles is proved by contradiction. Assume there exists a non-simple causation cycle $Y_1 = \langle y_0 y_1 \dots y_k y_1 \rangle^t$ where $y_0 = y_k$. From Lemma 1, $y_0 \prec_{\S} y_1$. However a new causation cycle Y_2 exists where: $Y_2 = \langle y_1 y_2 \dots y_{k-1} y_k y_1 \rangle^{t+1}$. Thus by Lemma 1, $y_1 \prec_{\S} y_k = y_0$ which is a contradiction. ■

The theoretical results in this section are the proofs for the three principles of safe policy routing dynamics introduced in Section II. The non-interference principle comes from Theorem 1, which states that every causation chain in an economic DPR instance must be valley-free. The single and multi-tiered cycle principles come from Theorem 2, which states that every causation cycle in an economic DPR instance is vertical and simple.

V. INTERFERENCEBEAT

In this section, we outline a distributed algorithm, INTERFERENCEBEAT, that checks if the principles of safe policy routing dynamics are maintained or whether policy violations exist. This is accomplished by detecting forbidden causation chains (including cycles) induced by policy violations. Once a forbidden causation chain is detected, the ASes involved need to collaborate to resolve the potential problem.

A. Description of INTERFERENCEBEAT

INTERFERENCEBEAT piggybacks a small token alongside route updates. When a node y receives a route update from its neighbor v at time t , it also receives a token θ_{in} . If node y selects a new path then it broadcasts a new token θ_{out} alongside

its own route update at time $t + 1$. Tokens are passed along causation chains. In general, a causation chain is started when a link flaps (*i.e.*, is lost or becomes available) or when a node changes its path preferences. A token consists of three parts, (i, r, n) . The identifier of the causation chain is i . The economic relationship between y and its predecessor v on the causation chain is $r \in \{\succ_{\$}, \prec_{\$}, =_{\$}, \emptyset\}$. For example, if v is a provider to y , then r is $\succ_{\$}$. The counter n keeps track of the number of times the token was passed along a customer-to-provider or a provider-to-customer link.

The PROCESS function, outlined in Figure 7, performs basic routing tasks and handles the incoming and outgoing tokens. It is invoked in every node y at time t after receiving all routing update messages. In steps 2 and 3, node y chooses and adopts its best available path. If y 's assigned path has changed in step 4 (*i.e.*, an action occurred), then node y 's causing neighbor v is identified in step 5. The contents of the token received from neighbor v are recovered in step 6. In step 7, the CREATETOKEN function is called which returns the contents of the new token to be sent out by y at time $t + 1$. The CHECKPRINCIPLES function is called in step 8. Node y stores information about the outgoing token in step 9. In step 10, the outgoing token created by node y is disseminated to all y 's neighbors.

```

1: function PROCESS( $y, t$ )
2:    $\text{Best}(y, t) \leftarrow \max_{\succ_t} \text{Choices}(y, t)$ 
3:    $\pi(y, t + 1) \leftarrow \text{Best}(y, t)$ 
4:   if  $\pi(y, t + 1) \neq \pi(y, t)$  then
5:      $v = C(y, t)$ 
6:      $\theta_{\text{in}} = \text{GETTOKENFROMNEIGHBOR}(y, v, t)$ 
7:      $\theta_{\text{out}} = \text{CREATETOKEN}(y, v, \theta_{\text{in}})$ 
8:      $\text{CHECKPRINCIPLES}(y, v, \theta_{\text{in}}, \theta_{\text{out}})$ 
9:      $\text{STORETOKEN}(y, v, \theta_{\text{out}})$ 
10:     $\text{SENDDTOKEN}(y, t, \theta_{\text{out}})$ 

```

Fig. 7. PROCESS function.

The CREATETOKEN function is outlined in Figure 8. Step 2 retrieves the needed parts from the incoming token. If the identifier i_{in} is empty in step 3 then a new one is generated in step 4. Otherwise, in step 6, the outgoing identifier i_{out} is set to the incoming identifier i_{in} . In step 7, the economic relationship between v and y is obtained and stored in r_{out} . In steps 8 through 11, the outgoing counter n_{out} is only incremented if nodes y and v are *not* peers. The outgoing token is returned in step 12.

The CHECKPRINCIPLES function is outlined in Figure 9. Steps 2 and 3 retrieve the needed parts from the tokens. Step 4 checks for the existence of a valley causation chain. If one is found, then interference is reported, where the causing node v , the chain identifier i_{in} and the relationship r_{in} are identified. In step 6, node y determines if it has previously received a token with identifier i_{in} . If so, then a cycle is detected. Node y recovers the old information in step 7. If the token was previously received from the same neighbor v then a non-simple cycle is reported in step 9. Step 10 checks if the token previously received contained the same counter value. If so,

```

1: function CREATETOKEN( $y, v, \theta_{\text{in}}$ )
2:    $(i_{\text{in}}, \_, n_{\text{in}}) = \theta_{\text{in}}$ 
3:   if  $i_{\text{in}}$  is  $\emptyset$  then
4:      $(i_{\text{out}}, r_{\text{out}}, n_{\text{out}}) = (\text{NEWID}(), \emptyset, 0)$ 
5:   else
6:      $i_{\text{out}} = i_{\text{in}}$ 
7:      $r_{\text{out}} = \text{ECONOMICRELATION}(v, y)$ 
8:     if  $r_{\text{out}}$  is equal to  $=_{\$}$  then
9:        $n_{\text{out}} = n_{\text{in}}$ 
10:    else
11:       $n_{\text{out}} = n_{\text{in}} + 1$ 
12:    return  $(i_{\text{out}}, r_{\text{out}}, n_{\text{out}})$ 

```

Fig. 8. CREATETOKEN function.

then the token was only passed between peers since leaving node y and a horizontal cycle is reported in step 11.

```

1: function CHECKPRINCIPLES( $y, v, \theta_{\text{in}}, \theta_{\text{out}}$ )
2:    $(i_{\text{in}}, r_{\text{in}}, \_) = \theta_{\text{in}}$ 
3:    $(\_, \_, n_{\text{out}}) = \theta_{\text{out}}$ 
4:   if  $(r_{\text{in}}$  is equal to  $\succ_{\$}$  or  $=_{\$}$ ) and  $(v \preceq_{\$} y)$  then
5:      $\text{REPORTINTERFERENCE}(y, v, \theta_{\text{in}})$ 
6:   if  $\text{HASRECEIVEDTOKEN}(y, i_{\text{in}})$  then
7:      $(v_{\text{old}}, n_{\text{old}}) = \text{GETSTOREDTOKEN}(y, i_{\text{in}})$ 
8:     if  $v_{\text{old}}$  is equal to  $v$  then
9:        $\text{REPORTNONSIMPLECYCLE}(y, v, \theta_{\text{in}})$ 
10:    if  $n_{\text{old}}$  is equal to  $n_{\text{out}}$  then
11:       $\text{REPORTEHORIZONTALCYCLE}(y, v, \theta_{\text{in}})$ 

```

Fig. 9. CHECKPRINCIPLES function.

B. Sample Operation of INTERFERENCEBEAT

Figure 10 shows the operation of INTERFERENCEBEAT on the DPR instance described in Figure 3, assuming y_0, y_1 and y_2 are all peers. At time $t + 1$, node y_0 initiates a new causation chain with identifier ID1 and sends a token to y_1 . Since y_0 initiated the chain, the count is 0 and the relationship is \emptyset . Node y_1 takes an action and sends a new token to y_2 . Since y_1 and y_0 are peers, the relationship is set to $=_{\$}$ and the count is still 0 as the token only traversed a peering link. Finally, since y_2 is a peer to its causing node y_1 , interference is detected by y_2 upon receiving the token.

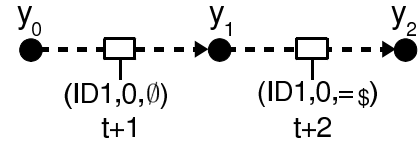


Fig. 10. Sample operation of INTERFERENCEBEAT.

C. Properties of INTERFERENCEBEAT

INTERFERENCEBEAT has the following characteristics:

- **Efficient Space.** A small token of space complexity $O(1)$ (a few bytes) is appended to each routing update message

irrespective of how the routing dynamics manifest in the network.

- **Provably Correct.** INTERFERENCEBEAT is based on a comprehensive theory of policy routing dynamics and hence is provably correct with any dynamic network. In other words, any changes in network topology or path preferences do not affect the correctness of detecting policy violations.
- **Adoptable.** INTERFERENCEBEAT enables results even in the case of gradual adoption of the protocol. To detect policy violations, only the ASes along the causation chain need to adopt the protocol. Thus neighboring ASes can use INTERFERENCEBEAT to detect misconfigurations.
- **Privacy Preserving.** ASes only reveal information to their immediate neighbors and local policy information is not explicitly shared.

D. Practical Considerations for INTERFERENCEBEAT

INTERFERENCEBEAT could be implemented over BGP where the token is passed in the message options. When an AS initiates a new causation chain it must create a new identifier using the NEWID() function. This can be accomplished by hashing the AS number, router identifier, time and destination prefix. A fixed number of bits can be allocated to the identifier, with more bits reducing the probability of a hash collision.

In INTERFERENCEBEAT, if a cycle or valley is detected by a node y , only its causing neighbor node v can be immediately identified. In order to identify/notify other nodes along the chain, a back-propagating alert protocol may be used. Each node can leverage its stored tokens to find its previous causing neighbor.

In appendix F we show that the synchronicity of DPR is not a hindrance and that it has sufficient expressive power to model asynchronicity. Hence, INTERFERENCEBEAT can be trivially extended to a real-time setting.

VI. VIOLATIONS OF THE ECONOMIC DPR MODEL

We formally define four common policy violations, which are relaxations of the economic DPR model. For each violation we prove the invariant properties of the resultant causation chains and cycles. The modelled dynamics induced by each violation can be compared against the dynamics observed by INTERFERENCEBEAT. If a violation cannot cause the observed behavior, then it can be ruled out.

A. Description of Violations





To describe paths and causation chains in better detail we categorize valleys into four subtypes.

Definition 20 (Valley Types). We extend definition 16 of valleys to four subtypes as shown in Table II.

Violation 1: Non-Strict Economic Relationships

With non-strict economic relationships, a node can be both a (direct or indirect) provider and a (direct or indirect) peer to another node. Figure 11 shows a comparison between non-strict and strict economic relationships.

TABLE II
VALLEY TYPES GIVEN SEQUENCE $\langle a b c \rangle$.

Valley Type	Condition	Illustration
\mathcal{A}	$a \succ_{\S} b \prec_{\S} c$	
\mathcal{B}	$a \succ_{\S} b =_{\S} c$	
\mathcal{C}	$a =_{\S} b \prec_{\S} c$	
\mathcal{D}	$a =_{\S} b =_{\S} c$	

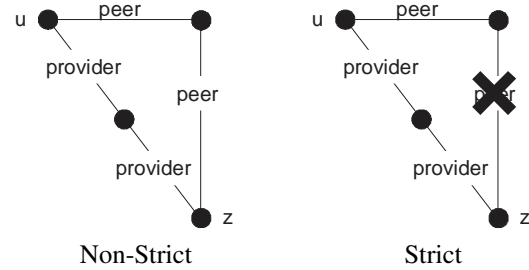


Fig. 11. Strict and non-strict economic relationships. In the strict variant, node u cannot be an indirect provider and peer to node z .

Violation 2: Transiting Between Peers

Generally, an AS only carries traffic that is destined to (or originating from) one of its customers. However, due to misconfigurations or complex agreements between peers, an AS may transit traffic between its peers. Economic DPR instances with this violation have an enlarged set of realizable paths. Paths containing valleys of type \mathcal{D} can be adopted by nodes. However, paths are forbidden if they contain valley types \mathcal{A} , \mathcal{B} , or \mathcal{C} . Therefore, every realizable path consists of a series of zero or more ascending customer-to-provider edges, followed by zero or more peer edges, followed by zero or more descending provider-to-customer edges, as shown in Figure 12.

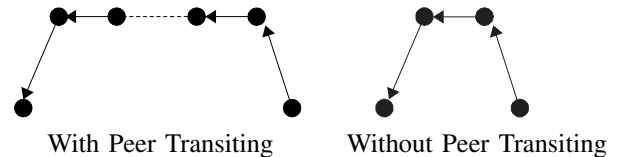


Fig. 12. Allowable paths in economic DPR with and without violation 2.









Violation 3: Peer Paths over Customer Paths

Whereas violation 2 is a relaxation on the set of realizable paths, violation 3 is a relaxation of the path preferences. Nodes in economic DPR instances with violation 3 can prefer peer paths over customer paths. Nodes, however, cannot prefer provider paths over peer/customer paths. Only valley-free paths are realizable.

Violation 4: Provider Paths over Peer/Customer Paths

Nodes in economic DPR instances with violation 4 can prefer provider paths over peer/customer paths. Again, only valley-free paths are realizable.

TABLE III
VIOLATIONS OF THE ECONOMIC DPR MODEL

Violation	Valley Types in Causation Chains:				Vertical Cycles	Horizontal Cycles	Potentially Unsafe?
	\mathcal{A}	\mathcal{B}	\mathcal{C}	\mathcal{D}			
0: None					simple	none	no
1: Non-Strict Economics					simple	none	no
2: Transiting					simple	non-simple, simple	yes
3: Peers Preferred					simple	non-simple, simple	yes
4: Providers Preferred					non-simple, simple	none	yes

B. Dynamics Induced by Violations

The four violations describe different variants of the economic DPR model. Each variant results in different types of causation chains and cycles. For ease of exposition, we model the resulting dynamics of each violation in isolation. The theoretical proofs for the violations can be found in the appendices

Table III summarizes the effects of each violation on the characteristics of causation chains and cycles. The first and second rows show the strict and non-strict economic DPR models. They are the only two variants guaranteed to be safe. The non-strict economic DPR model, however, when combined with other violations could lead to potentially unsafe behavior. The three other violations induce routing behavior which is potentially unsafe.

INTERFERENCEBEAT can be extended using the results of Table III. Upon the detection of a valley in the causation chain, its type (\mathcal{A} , \mathcal{B} , \mathcal{C} , or \mathcal{D}) can rule out possible causing violations. For example, if a valley of type \mathcal{B} was detected using INTERFERENCEBEAT, then violations 1, 2, and 3 can be immediately ruled out as the possible causes for the observed behavior. Similar methods can be used upon detection of causation cycles.

VII. RELATED WORK

Static models for BGP, such as the Stable Paths Problem (SPP) [4], provide insight into the steady-state behavior of policy routing. There are also offline methods that leverage SPP and utilize information from BGP tables [8] to infer policy conflicts between ASes. DPR extends SPP to give insight into the real-time transient behavior of networks. DPR allows us to reason about issues such as misconfigured routing policies or networks with sporadic link failures.

The canonical solution for detecting policy conflicts based on SPP is the Safe Path Vector Protocol (SPVP) introduced by Griffin *et al.* in [9]. SPVP exchanges route flaps among ASes in extended “history” messages. INTERFERENCEBEAT extends SPVP by appending additional information in a small token to each routing update message to detect violations of the Gao-Rexford guidelines [5].

There are many algorithms that attempt to detect and resolve policy conflicts. Counting [10] and other token-based [11] heuristic approaches benefit from having a low communication overhead. INTERFERENCEBEAT extends such approaches by leveraging the DPR model to guarantee the correctness of policy violation detection and diagnosis.

Finally, there are routing architectures that constrain traditional policy routing to guarantee convergence. Metarouting [12] defines a policy language based on a routing algebra that gives compile-time guarantees for routing convergence. In [13], real-time enforcement of convergence is achieved by passing information in tokens to affect policy rankings. INTERFERENCEBEAT does not enforce convergence. Instead it leverages the DPR model to detect non-compliance to the principles of safe routing dynamics and notifies ASes upon the detection of policy violations.

VIII. CONCLUSIONS

We introduced a Dynamic Policy Routing (DPR) model, which extends the static model of BGP to capture the *propagation dynamics* of route flaps due to *arbitrary* changes in topology or path preferences. The theoretical results of this paper can be summarized by three key principles which distill the properties of routing dynamics in a safe (economic) policy configuration.

We introduce INTERFERENCEBEAT, a novel distributed algorithm to detect and diagnose policy violations. INTERFERENCEBEAT has a beneficial set of characteristics such as efficiency, privacy, and adoptability. Diagnosis is further enhanced by modelling common policy violations such as the preference of peer paths over customer paths.

IX. ACKNOWLEDGMENT

This work has been partially supported by National Science Foundation awards: CISE/CCF #0820138, CISE/CSR #0720604, CISE/CNS #0524477, CNS/ITR #0205294, and CISE/EIA RI #0202067.

REFERENCES

- [1] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, “Delayed Internet Routing Convergence,” *IEEE/ACM Trans. Netw.*, vol. 9, pp. 293–306, June 2001.
- [2] A. Feldmann, O. Maennel, Z. Mao, A. Berger, and B. Maggs, “Locating Internet Routing Instabilities,” in *ACM SIGCOMM*, September 2004.
- [3] K. Varadhan, R. Govindan, and D. Estrin, “Persistent Route Oscillations in Inter-domain Routing,” *Computer Networks*, Tech. Rep., 1996.
- [4] T. Griffin, F. Shepherd, and G. Wilfong, “The Stable Paths Problem and Interdomain Routing,” *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 232–243, Apr 2002.
- [5] L. Gao and J. Rexford, “Stable Internet Routing Without Global Coordination,” *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, 2001.
- [6] D. Obradovic, “Real-time Model and Convergence Time of BGP,” in *INFOCOM*, 2002.
- [7] N. Feamster, H. Balakrishnan, and J. Rexford, “Some Foundational Problems in Interdomain Routing,” in *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004.

- [8] F. Wang and L. Gao, "Inferring and Characterizing Internet Routing Policies," in *ACM IMC*, 2003, pp. 15–26.
- [9] T. Griffin and G. T. Wilfong, "A Safe Path Vector Protocol," in *INFOCOM*, 2000, pp. 490–499.
- [10] J. Cobb and R. Musunuri, "Enforcing Convergence in Inter-domain Routing," *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, vol. 3, pp. 1353–1358 Vol.3, Nov.-3 Dec. 2004.
- [11] S. Yilmaz and I. Matta, "An Adaptive Management Approach to Resolving Policy Conflicts," in *IFIP Networking 2007*, Atlanta, Georgia, May 2007.
- [12] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 1–12.
- [13] C. T. Ee and V. Ramachandran and B.G. Chun and K. Lakshminarayanan and S. Shenker, "Resolving Inter-Domain Policy Disputes," in *SIGCOMM*, 2007, pp. 157–168.

APPENDIX A PROOF OF THEOREM 1

For convenience of notation, we drop the time index of certain terms with respect to a given chain $Y = \langle y_0 y_1 \dots y_k \rangle^t$, namely:

$$\begin{aligned}
 \pi(y_i) &= \pi(y_i, t+i) \\
 \pi_{\text{next}}(y_i) &= \pi(y_i, t+i+1) \\
 \rho(y_i) &= \rho(y_i, t+i) \\
 \rho_{\text{next}}(y_i) &= \rho(y_i, t+i+1) \\
 \text{RankDec}(y_i) &\text{ iff } \text{RankDec}(y_i, t+i) \\
 \text{RankSame}(y_i) &\text{ iff } \text{RankSame}(y_i, t+i) \\
 \text{RankInc}(y_i) &\text{ iff } \text{RankInc}(y_i, t+i)
 \end{aligned}$$

Theorem 1. Every causation chain of an economic DPR instance $(\succeq_{\S}, \succeq^t, G)$ is valley-free.

Proof: Assume not. Then there exists a causation chain $Y = \langle y_0 y_1 \dots y_k \rangle^t$ and an index i such that $0 < i < k$ and $y_{i-1} \succeq_{\S} y_i \preceq_{\S} y_{i+1}$. Thus y_{i-1} and y_{i+1} are peers or providers to y_i .

The first part of this proof shows that if this is the case, then at no time during the causation chain did y_i have a customer path. The second part of this proof shows that sometime during the causation chain y_{i+1} had a path through y_i . Therefore y_{i+1} had a realized valley path since y_i did not have a customer path and y_i is a customer of or peer to y_{i+1} . Since valley-paths are forbidden in economic DPR instances, this results in a contradiction. Since $C(y_i) = y_{i-1}$, either the first or second condition of causation from Table I holds for y_i at time $t+i$.

Case: y_i Causation Condition 1

If the first condition of Table I holds for y_i then: $\rho(y_i) = y_{i-1}$ and $\text{RankDec}(y_i)$, as shown in Figure 13. Therefore $\pi(y_i) \succ^{t+i} \pi_{\text{next}}(y_i)$. Let $v = \rho_{\text{next}}(y_i)$. It cannot be that $v \prec_{\S} y_i$. Otherwise, since $\pi_{\text{next}}(y_i)$ is a customer path and $\pi(y_i)$ is not a customer path (since $\rho(y_i) = y_{i-1} \succeq_{\S} y_i$), by the conditions of economic DPR instances: $\pi(y_i) \prec^{t+i} \pi_{\text{next}}(y_i)$, causing a contradiction as shown in Figure 14. Thus $v \succeq_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succeq_{\S} y_i$.

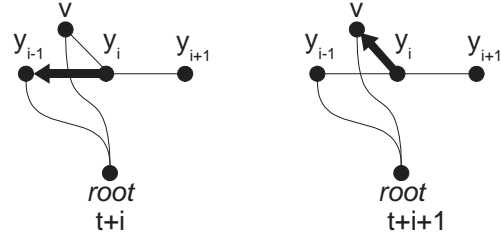


Fig. 13. Causation condition 1: $\text{RankDec}(y_i)$

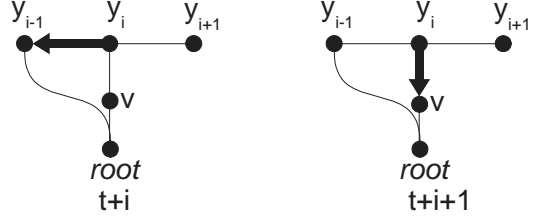


Fig. 14. Contradiction: $\text{RankInc}(y_i)$

Case: y_i Causation Condition 2

If the second condition of Table I holds for y_i then: $\rho_{\text{next}}(y_i) = y_{i-1}$ and $\text{RankInc}(y_i)$, as shown in Figure 15. Therefore $\pi(y_i) \prec^{t+i} \pi_{\text{next}}(y_i)$. Let $v = \rho(y_i)$. It cannot be that $v \prec_{\S} y_i$. Otherwise, since $\pi(y_i)$ is a customer path and $\pi_{\text{next}}(y_i)$ is not (since $\rho_{\text{next}}(y_i) = y_{i-1} \succeq_{\S} y_i$), by the conditions of economic DPR instances: $\pi(y_i) \succ^{t+i} \pi_{\text{next}}(y_i)$, causing a contradiction, as shown in Figure 16. Thus $\rho_{\text{next}}(y_i) \succeq_{\S} y_i$ and $v \succeq_{\S} y_i$. So for both cases, at no time in the causation chain did y_i have a customer path:

$$\rho(y_i) \succeq_{\S} y_i \text{ and } \rho_{\text{next}}(y_i) \succeq_{\S} y_i$$

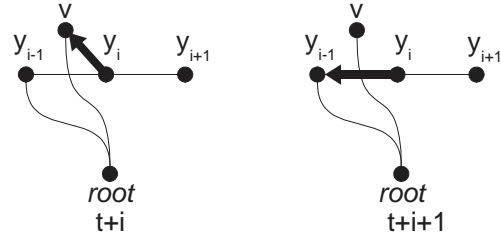


Fig. 15. Causation condition 2: $\text{RankInc}(y_i)$

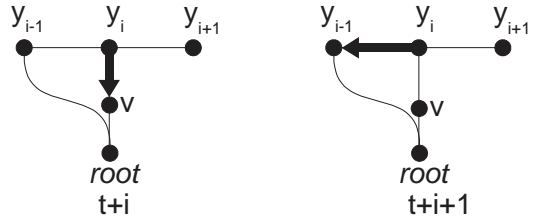


Fig. 16. Contradiction: $\text{RankDec}(y_i)$

Case: y_{i+1} Causation Condition 1

If the first causation condition of Table I holds for y_{i+1} ,

then $\rho(y_{i+1}) = y_i$. By Proposition 1: $\pi(y_{i+1}) = \langle y_{i+1} \pi(y_i) \rangle$. $\pi(y_{i+1})$ is a valley path since $y_{i+1} \succeq_{\S} y_i \preceq_{\S} \rho(y_i)$. Since all valley paths are forbidden, $\pi(y_{i+1})$ can never be realized, causing a contradiction.

Case: y_{i+1} Causation Condition 2

Similar arguments can be used if the second causation condition of Table I holds for y_{i+1} : $\rho_{\text{next}}(y_{i+1}) = y_i$. Thus by Proposition 1: $\pi_{\text{next}}(y_{i+1}) = \langle y_{i+1} \pi_{\text{next}}(y_i) \rangle$. $\pi_{\text{next}}(y_{i+1})$ is a valley path since $y_{i+1} \succeq_{\S} y_i \preceq_{\S} \rho_{\text{next}}(y_i)$, and can never be realized. Thus in all cases a contradiction occurs, proving the theorem. ■

APPENDIX B

THEOREMS AND PROOFS FOR VIOLATION 1

Violation 1 involves the most complicated constructions. The following subsection formally defines non-strict economic relationships. If an economic DPR has non-strict economic relationships $D = (\succeq_*, \preceq_*, G)$, then it contains the economic operator \succeq_* . From \succeq_* , a tight economic relation is defined by:

$$u \succ_* v \text{ iff } u \succeq_* v \text{ and } u \not\prec_* v$$

and no relation is defined by:

$$u \parallel_* v \text{ iff } u \not\prec_* v \text{ and } u \not\succeq_* v$$

The customer, peer, and provider economic relationships can be derived from the operator \succeq_* :

- If u is a customer of v , then $u \prec_* v$.
- If u is a provider to v , then $u \succ_* v$.
- If u is a peer to v , then $u \parallel_* v$.

The transitive properties of the economic operator \succeq_* can be modeled using post-order conditions:

- 1) (reflexive) $x \succeq_* x$
- 2) (anti-symmetric) $x \succeq_* y$ and $y \succeq_* x$ implies $x = y$
- 3) (transitive) $x \succeq_* y$ and $y \succeq_* z$ implies $x \succeq_* z$

The key difference between a strict and non-strict economic operator is that peering relationships are not transitive in the non-strict variant. Whereas peering is represented by the equivalence relation $=_{\S}$ in the strict variant, peering is represented by no relation \parallel_* in the non-strict variant. Thus as shown in figure 17, strict economic relationships form equivalence classes with the peering relation $=_{\S}$, which are not present in standard economic relationships. This enables a node to be both an indirect peer and provider to another node in the standard variant. However it should be noted that provider-to-customer relationships are transitive in both variants.

For ease of notation, the following notation is used to describe that node x is a peer or provider to node y :

$$x \approx_* y \text{ iff } x \not\prec_* y$$

We define paths by the economic relationship between a path's starting node u and its next-hop. For all paths P^u :

$$\begin{aligned} \text{Customer}(P^u) &\Leftrightarrow u \succ_* \text{NextHop}(P^u) \\ \text{Peer}(P^u) &\Leftrightarrow u \parallel_* \text{NextHop}(P^u) \\ \text{Provider}(P^u) &\Leftrightarrow u \prec_* \text{NextHop}(P^u) \end{aligned}$$

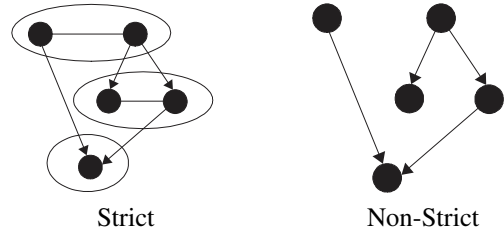


Fig. 17. Strict and Non-Strict economic relationships. The circles over the nodes in the strict variant represent equivalent classes of peers.

Given a sequence of nodes $\langle a b c \rangle$, valley types are represented as follows:

Valley Type	Condition	Illustration
\mathcal{A}	$a \succ_* b \prec_* c$	
\mathcal{B}	$a \succ_* b \parallel_* c$	
\mathcal{C}	$a \parallel_* b \prec_* c$	
\mathcal{D}	$a \parallel_* b \parallel_* c$	

Theorem 3. All causation chains of non-strict economic DPR instances are valley-free.

Proof: The proof follows exactly as the proof for theorem 1, only by replacing the \succ_{\S} with \succ_* and \preceq_{\S} with \preceq_* . ■

Theorem 4. All causation cycles of non-strict economic DPR instances are vertical and simple.

Proof: Let $Y = \langle y_0 y_1 \dots y_k \rangle^t$ be a causation cycle, where $y_0 = y_k$. The cases for this proof can be partitioned by y_1 's economic relationship with y_0 :

Case (a): $y_0 \succ_* y_1$:

If $y_0 \succ_* y_1$, since Y is valley-free, $y_i \succ_* y_{i+1}$ for $0 \leq i < k$. However $y_0 \succ_* y_k = y_0$, causing a contradiction and eliminating this case.

Case (b): $y_0 \parallel_* y_1$:

If $y_0 \parallel_* y_1$, since Y is valley-free, $y_i \succ_* y_{i+1}$ for $1 \leq i < k$. Thus Y is vertical. Y has to be simple, otherwise $\langle y_{k-1} y_0 y_1 \rangle$ would be a realized causation chain. Since $y_{k-1} \succ_* y_0$ and $y_0 \parallel_* y_1$, the causation chain is a valley, causing a contradiction. Therefore Y is simple and vertical.

Case (c): $y_0 \prec_* y_1$:

Assume $y_0 \prec_* y_1$. Thus Y is vertical. The cases can be further partitioned by y_{k-1} 's economic relationship with y_k . If $y_{k-1} \prec_* y_k$, then by the definition of valley-free sequences, $y_{i-1} \prec_* y_i$ for all $0 < i \leq k$. Thus $y_0 \prec_* y_k = y_0$, which is a contradiction. Therefore $y_{k-1} \approx_* y_k$. If Y is non-simple, then $\langle y_{k-1} y_0 y_1 \rangle$ would be a realized causation chain of. Since $y_{k-1} \approx_* y_0 = y_k$ and $y_0 \prec_* y_1$, the causation chain is a valley, causing a contradiction. Therefore Y is simple and vertical. ■

Remark 1. Non-strict economic follows instances are safe. This follows from the results of [5].

APPENDIX C

THEOREMS AND PROOFS FOR VIOLATION 2

Theorem 5. Every causation chain in an economic DPR instance with violation 2 does not admit valley types \mathcal{A} , \mathcal{B} or \mathcal{C} .

Proof: Assume not. Then there exists a causation chain $Y = \langle y_0 y_1 \dots y_k \rangle^t$ and an index i such that $0 < i < k$ and at least one of the two conditions hold:

- (a) $y_{i-1} \succ_{\S} y_i \preceq_{\S} y_{i+1}$
- (b) $y_{i-1} \preceq_{\S} y_i \succ_{\S} y_{i+1}$

Case (a): $y_{i-1} \succ_{\S} y_i \preceq_{\S} y_{i+1}$

If case (a) holds, then it can be shown that both $\rho(y_i) \succ_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succ_{\S} y_i$. This can be seen by looking at the causation conditions of y_i . If causation condition 1 holds for y_i , then $y_{i-1} = \rho(y_i)$ and $\text{RankDec}(y_i)$. It cannot be the case that $\rho_{\text{next}}(y_i) \preceq_{\S} y_i$, since this would imply that y_i switched from a provider path through y_{i-1} to a non-provider path, since $y_i \prec_{\S} \rho(y_i) = y_{i-1}$ and $y_i \succeq_{\S} \rho_{\text{next}}(y_i)$. This would imply $\text{RankInc}(y_i)$, causing a contradiction. Thus $\rho(y_i) \succ_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succ_{\S} y_i$. If causation condition 2 holds for y_i , then $y_{i-1} = \rho_{\text{next}}(y_i)$ and $\text{RankInc}(y_i)$. It cannot be the case that $\rho(y_i) \preceq_{\S} y_i$, since this would imply that y_i switched from a non-provider path to a provider path through y_{i-1} , since $y_i \succeq_{\S} \rho(y_i)$ and $y_i \prec_{\S} \rho_{\text{next}}(y_i) = y_{i-1}$. This would imply $\text{RankDec}(y_i)$, causing a contradiction. Thus for both cases, $\rho(y_i) \succ_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succ_{\S} y_i$.

Thus given the results above, we can prove that y_{i+1} had a realized path with valley type \mathcal{A} or \mathcal{C} . If causation condition 1 holds for y_{i+1} , then $\pi(y_{i+1}) = \langle y_{i+1} \pi(y_i) \rangle$. Since $y_{i+1} \succeq_{\S} y_i$ and $y_i \prec_{\S} \rho(y_i)$, then $\pi(y_{i+1})$ is a realized path with valley type \mathcal{A} or \mathcal{C} , causing a contradiction. If causation condition 2 holds for y_{i+1} , then $\pi_{\text{next}}(y_{i+1}) = \langle y_{i+1} \pi_{\text{next}}(y_i) \rangle$. Since $y_{i+1} \succeq_{\S} y_i$ and $y_i \prec_{\S} \rho_{\text{next}}(y_i)$, then $\pi_{\text{next}}(y_{i+1})$ is a realized path with valley type \mathcal{A} or \mathcal{C} , causing a contradiction.

Case (b): $y_{i-1} \preceq_{\S} y_i \succ_{\S} y_{i+1}$

If case (b) holds, then using an argument similar to case (a) it can be shown that both $\rho(y_i) \succeq_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succeq_{\S} y_i$. We can then prove that y_{i+1} had a realized path with valley type \mathcal{A} or \mathcal{B} , causing a contradiction. ■

Theorem 6. Every vertical causation cycle $Y = \langle y_0 \dots y_k \rangle^t$ in an economic DPR instance with violation 2 is simple.

Proof: This proof proceeds by determining y_1 's economic relationship with y_0 and y_{k-1} 's economic relationship with $y_k = y_0$. Since Y is a vertical causation cycle, there exists a minimal index i , $0 < i < k$ such that $y_i \not\equiv_{\S} y_{i-1}$. Note that $i \neq k$, otherwise $y_0 \equiv_{\S} y_1 \equiv_{\S} \dots \equiv_{\S} y_{k-1} \not\equiv_{\S} y_k$, implying $y_0 \not\equiv_{\S} y_k$, which is a contradiction. Either $y_i \succ_{\S} y_{i-1}$ or $y_i \prec_{\S} y_{i-1}$. It cannot be that $y_{i-1} \succ_{\S} y_i$, since by Theorem 5 $y_0 \equiv_{\S} y_{i-1} \succ_{\S} y_i \succ_{\S} y_{i+1} \dots \succ_{\S} y_k$, implying $y_0 \succ_{\S} y_k$ which is a contradiction. Therefore $y_{i-1} \prec_{\S} y_i$. If $i > 1$, then $y_{i-2} \equiv_{\S} y_{i-1} \prec_{\S} y_i$, representing a valley of type \mathcal{C} , which is a contradiction. So $i = 1$ and $y_0 \prec_{\S} y_1$.

Let j be the first index $1 < j < k$ where $y_{j-1} \succ_{\S} y_j$. Note that j has to exist otherwise $y_0 \prec_{\S} y_1 \preceq_{\S} \dots \preceq_{\S} y_k$,

implying $y_0 \prec_{\S} y_k$ which is a contradiction. From Theorem 5, $y_{h-1} \succ_{\S} y_h$ for all $j < h \leq k$. So $y_{k-1} \succ_{\S} y_k = y_0$. Therefore Y must be simple, otherwise $\langle y_{k-1} y_0 y_1 \rangle$ must be a causation chain. However since $y_{k-1} \succ_{\S} y_0$ and $y_0 \prec_{\S} y_1$, Y contains a valley of type \mathcal{A} , contradicting Theorem 5, and thus proving the theorem. ■

Theorem 7. An economic DPR instance with violation 2 admits simple and non-simple horizontal causation cycles.

Proof: From the example shown in Figure 18 which is identical to the ‘‘Bad Gadget’’ described in [4]. ■

Theorem 8. An economic DPR instance with violation 2 is potentially unsafe.

Proof: From the example shown in Figure 18, no stable assignment exists. ■

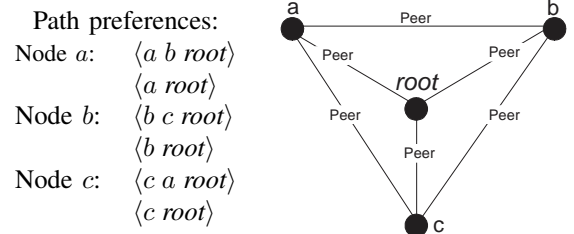


Fig. 18. Non-simple horizontal cycle for an economic DPR instance with violation 2. Paths not listed in the path preferences are forbidden.

APPENDIX D

THEOREMS AND PROOFS FOR VIOLATION 3

Theorem 9. Every causation chain in an economic DPR instance with violation 3 does not admit valley types \mathcal{A} or \mathcal{B} .

Proof: Assume not. Then there exists a causation chain $Y = \langle y_0 y_1 \dots y_k \rangle^t$ and an index i such that $0 < i < k$ and $y_{i-1} \succ_{\S} y_i \preceq_{\S} y_{i+1}$. The same reasoning as case (a) from the proof of Theorem 5 can be used. By considering the causation conditions of y_i , it can be shown that both $\rho(y_i) \succ_{\S} y_i$ and $\rho_{\text{next}}(y_i) \succ_{\S} y_i$. We can then prove that y_{i+1} had a realized path with valley type \mathcal{A} or \mathcal{B} , causing a contradiction. ■

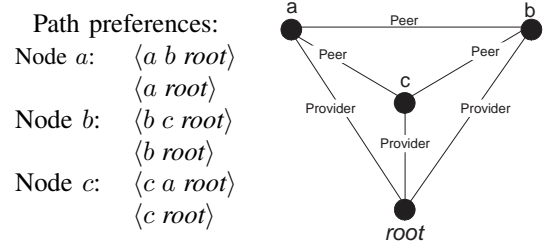


Fig. 19. Non-simple horizontal cycle for an economic DPR instance with violation 3. Paths not listed in the path preferences are forbidden.

Theorem 10. Every vertical causation cycle in an economic DPR instance with violation 3 is simple.

Proof: Assume not. Let vertical causation cycle $Y = \langle y_0 y_1 \dots y_k \rangle^t$ be non-simple. Since Y is a vertical causation

cycle, there exists a minimal index i , $0 < i < k$ such that $y_i \neq_{\$} y_{i-1}$. Following a similar argument as the one used to prove Theorem 6 we can prove that Y contains a valley of type \mathcal{A} or \mathcal{B} , which is a contradiction. ■

Theorem 11. *An economic DPR instance with violation 3 admits simple and non-simple horizontal causation cycles.*

Proof: From the example shown in Figure 19. This example is identical to the “Bad Gadget” described in [4]. ■

Theorem 12. *An economic DPR instance with violation 3 is potentially unsafe.*

Proof: From the example shown in Figure 19, no stable assignment exists. ■

APPENDIX E

THEOREMS AND PROOFS FOR VIOLATION 4

Theorem 13. *Every causation chain in an economic DPR instance with violation 4 does not admit valley types \mathcal{C} or \mathcal{D} .*

Proof: Assume not. Then there exists a causation chain $Y = \langle y_0 y_1 \dots y_k \rangle^t$ and an index i such that $0 < i < k$ and:

$$y_{i-1} =_{\$} y_i \preceq_{\$} y_{i+1}$$

The rest of the proof follows similarly to that of theorem 5. First it is shown that both $\rho(y_i) \succeq_{\$} y_i$ and $\rho_{\text{next}}(y_i) \succeq_{\$} y_i$. Then it is shown that either $\rho(y_{i+1})$ or $\rho_{\text{next}}(y_{i+1})$ is a valley path, causing a contradiction. ■

Theorem 14. *There are no horizontal cycles in economic DPR instances with violation 4.*

Proof: This follows directly from theorem 13, which states that causation chains of type \mathcal{D} do not exist. ■

Theorem 15. *An economic DPR instance with violation 4 admits simple and non-simple vertical causation cycles.*

Proof: From the example shown in Figure 20. This example is identical to the “Bad Gadget” described in [4]. ■

Path preferences:
Node a: $\langle a b \text{ root} \rangle$
 $\langle a \text{ root} \rangle$
Node b: $\langle b c \text{ root} \rangle$
 $\langle b \text{ root} \rangle$
Node c: $\langle c a \text{ root} \rangle$
 $\langle c \text{ root} \rangle$

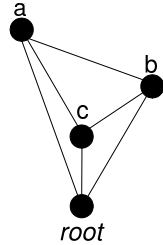


Fig. 20. Non-simple horizontal cycle for an economic DPR instance with violation 4. All edges are customer/provider links. Paths not listed in the path preferences are forbidden.

Theorem 16. *An economic DPR instance with violation 4 is potentially unsafe.*

Proof: From the example shown in Figure 20, no stable assignment exists. ■

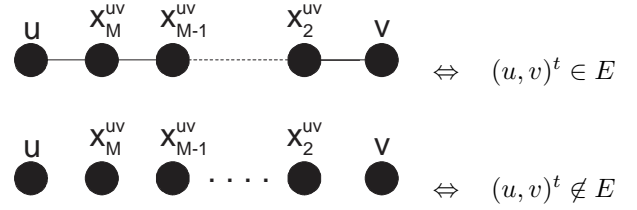


Fig. 21. Transit Nodes

APPENDIX F

ASYNCHRONICITY WITH DPR

This section describes how the DPR model can simulate asynchronicity. We assume that we have a regular DPR instance $D = (\succeq, G)$ which we wish to augment with asynchronicity. There are several ways to represent asynchronicity. We will use link delays. This choice enables us to use the existing DPR model without adding new constructs. At any time t , each link $(u, v)^t \in E$ admits a variable time delay between 1 and a finite upper limit M .

This delay is specified by the function $L(u, v, t)$ which outputs an integer in $[1, M]$. The time delays are considered ordered, such that $L(u, v, t) - L(u, v, t + k) < k$. Thus the values $L(u, v, 4) = 100$ and $L(u, v, 5) = 2$ are not allowed since v would get u 's path at time 5 before receiving u 's path at time 4. From DPR instance D and delay function L , a new DPR instance $D' = (\succeq', G')$ can be constructed to simulate D with the time delays.

For every pair of nodes in the original instance D , a set of $M - 1$ transit nodes will be added to D' . These transit nodes represent the “communication wire” between every two nodes. The dynamic nature of the links in DPR instances will be used to control the length of the “communication wire”. If $L(u, v, t) = 5$, then a path of length 5 between u and v through the transit nodes will appear at time t .

A. Graph of Asynchronous DPR Instances

For every node u in the original DPR instance D , there is a corresponding node in the asynchronous DPR instance D' :

$$u \in V \Rightarrow u \in V'$$

For every two nodes u, v in D , there are $M - 1$ transit nodes:

$$u, v \in V \Rightarrow x_i^{uv} \in V' \text{ for } 2 \leq i \leq M$$

Each transit node is connected to its neighbors. This connection forms the longest possible communication between nodes u and v . It toggles on/off with the connectivity of $(u, v)^t \in E$ for each time t , as shown in figure 21.

$$\left\{ \begin{array}{l} (u, x_M^{uv})^t \in E' \\ (x_{i+1}^{uv}, x_i^{uv})^t \in E' \text{ for all } 1 < i < M \\ (x_2^{uv}, v)^t \in E' \end{array} \right\} \text{ iff } (u, v)^t \in E$$

The time delays $L(u, v, t)$ describe the “shortcut” available through the transit nodes at each time t :

$$\begin{aligned} (u, x_i^{uv})^t \in E' & \text{ iff } (u, v)^t \in E \text{ and } L(u, v, t) = i \\ (u, v)^t \in E' & \text{ iff } (u, v)^t \in E \text{ and } L(u, v, t) = 1 \end{aligned}$$

An example of a delay of one and three between nodes u and v can be seen in figures 22 and 23.

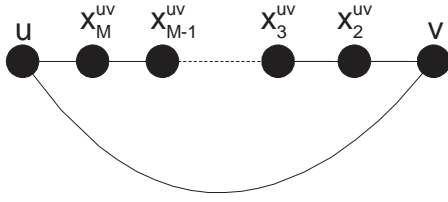


Fig. 22. Transit nodes simulating a delay of $L(u, v, t) = 1$.

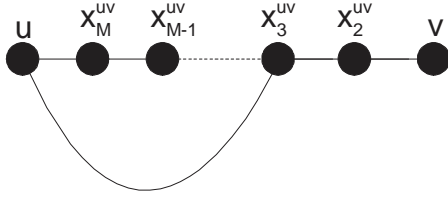


Fig. 23. Transit nodes simulating a delay of $L(u, v, t) = 3$.

B. Path Preferences of Asynchronous DPR Instances

The path preferences of the asynchronous DPR: $D' = (\succeq', G')$ discount the presence of transit nodes in paths. Let the operation `RemoveTransit` remove all transit nodes of a sequence. This operation allows us to derive the asynchronous path preferences from the original synchronous path preferences. Thus for all non-transit nodes $u \in V'$:

$$P_1^u \succeq'^t P_2^u \text{ iff } \text{RemoveTransit}(P_1^u) \succeq^t \text{RemoveTransit}(P_2^u)$$

Each transit node x_i^{uv} prefers a path through its source node u than through its transit neighbor toward the source: x_{i+1}^{uv} . Paths containing sequences in the opposite direction of the “communication link” (from x_i^{uv} to x_{i-1}^{uv}) are forbidden.

C. Redundant Connections

The transformation from synchronous to asynchronous DPR instances described above needs to be enhanced to avoid transient routing losses. This can occur during abrupt changes in connection delays as shown in figure 24.

In order to remedy this situation, redundant links between the source node u and the transit nodes are established, as shown in figure 25. This enables path consistency during changes of communication delays. Thus the proper transformation of links from synchronous D to asynchronous D' can be represented as:

$$\begin{aligned} (u, x_i^{uv})^t \in E' & \text{ iff } (u, v)^t \in E \text{ and } L(u, v, t) \leq i \\ (u, v)^t \in E' & \text{ iff } (u, v)^t \in E \text{ and } L(u, v, t) = 1 \end{aligned}$$

D. Causation Chains in Asynchronous DPR Instances

The definition of causation chains is not changed for asynchronous DPR instances. Given delay $L(u, v, t) = 3$, a causation chain of $\langle u v \rangle^t$ in the original DPR instance D would correspond to a causation chain of $\langle u x_3^{uv} x_2^{uv} v \rangle^t$ in the asynchronous DPR instance D' .

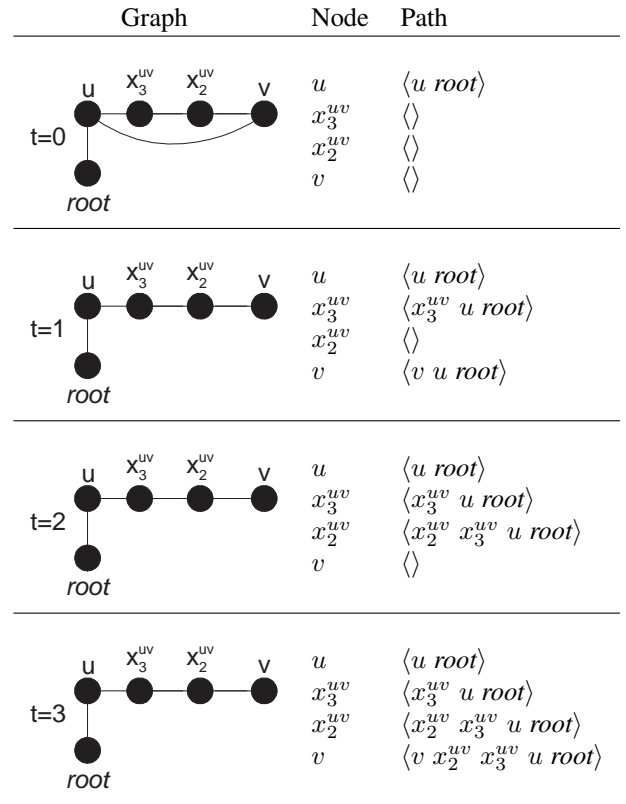


Fig. 24. Node v has a transient path loss from node u . This is due to an increase in delay from $L(u, v, 0) = 1$ to $L(u, v, 1) = 3$.

E. Asynchronous Economic DPR Instances

Asynchronous economic DPR instances can follow the definitions described in Section IV. Transit nodes have no economic relationships with the other nodes. The domain of the economic operator $\succeq_{\mathcal{S}}$ is only over non-transit nodes. Characterization of sequences (causation chains or paths) is accomplished by using the `RemoveTransit` operator. A path P in D' is valley-free if its corresponding transit-free path `RemoveTransit`(P) is valley-free. Similarly, a causation chain Y in D' is valley-free if its corresponding transit-free chain `RemoveTransit`(Y) is valley-free. Similar use of `RemoveTransit` can be employed to characterize customer, peer, and provider paths. From this construction, the proofs of this report are unchanged except for the application of the `RemoveTransit` operator.

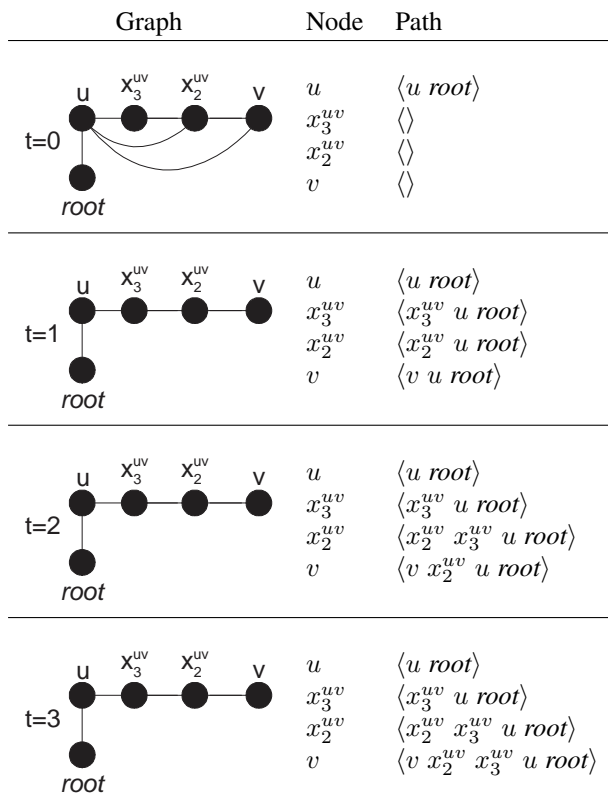


Fig. 25. An increase in connection delay occurred from $L(u, v, 0) = 1$ to $L(u, v, 1) = 3$. Transient path loss at node v is prevented due to redundant connections.