Manual: Network Contributor Rewards

Nihar Shah

DoubleZero Foundation

May 1, 2025

1 Overview

This manual explains the DoubleZero rewards model for network contributors, which is a mechanism that uses **Shapley values** to distribute rewards among network contributors (e.g. private link operators) in proportion to the *marginal value* each contributor adds to the network's efficiency. This is operationalized in `network_shapley.py` at https://github.com/nihar-dz/network-shapley.

Users who only wish to use the code should proceed directly to the Github link that contains the core code, a short readme file that explains how to run the code, and some example inputs. This longer manual is for readers who want to understand the complexities and theory behind the approach.

- Section 2 motivates the choice of Shapley values over the carried traffic model to reward network contributors.
- Section 3 walks through a minimal model with three nodes and three links.
- Section 4 layers on four complications: operator withdrawals, non-contiguous networks, latency measurements, and future trajectories of demand.
- Section 5 introduces the full model and describes some simulation results.

2 Motivating Shapley Values

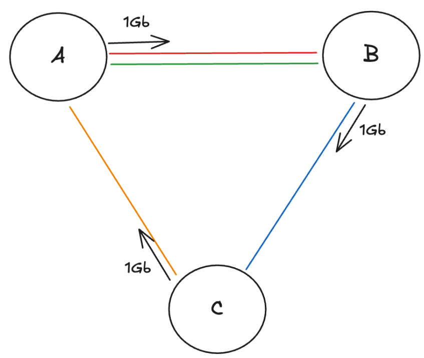
The rewards model is the key to making the DoubleZero network useful in a decentralized manner. At a high level, the rewards model accumulates revenue from users and distributes that to network contributors (net of rewards removed to prevent over-concentration or inorganic traffic attacks). But the critical question is how rewards are distributed *amongst* contributors.

The rewards model uses the concept of **Shapley values** from the field of cooperative game theory to do so. This framework gives rewards to a network contributor *fairly* and in proportion to its *marginal* contribution to the global value function.

Shapley values may appear to be needlessly complex in favor of a simpler approach, which is to pay out rewards on the basis of carried traffic. However, we believe the carried traffic model – while easier to understand – is insufficiently discriminating and does not incentivize long-term value.

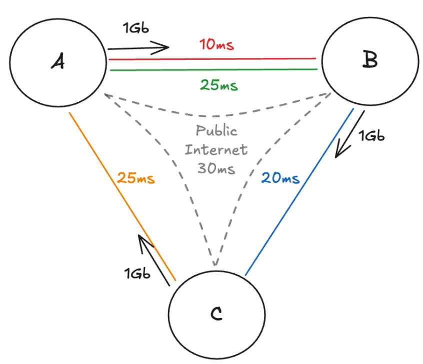
To illustrate the point, consider a scenario with four links connecting cities A, B, and C, depicted in Figure 1. The red and green links connect A and B; an orange link connects A and C; and a blue link connects B and C. Suppose the red link is more performant than the green link. Finally, each city wants to send one gigabit of traffic to its clockwise neighbor. Under this topology and using the carried traffic model, the division of rewards is straightforward: the green link receives no rewards (as it carries no traffic) while each of the other links receives one-third of the rewards (as they each carry one gigabit).

Figure 1: Three-City Example



But these rewards do not map to true value created. For instance, consider the world without each of the links sequentially. Should the red link drop out, the green link becomes much more important. Should the orange or blue links drop out, users will care if the public internet pathway connecting those cities is meaningfully worse; but they will not care if the performance is comparable. Indeed, to augment the scenario, suppose the public internet pathway connecting each city has a one-way latency of 30ms. Suppose the red line is substantially more performant than the green line (with a latency of 10ms rather than 25ms); and that the blue line is more performant (20ms) than the orange line (25ms). This is depicted in Figure 2.

Figure 2: Three-City Example (Detailed)



With this added information and under the lens of counterfactual scenarios, the division of rewards on the basis of traffic carried no longer seems optimal. The green link should get more than zero for its insurance value; the blue link should earn more rewards than the orange link when benchmarked against the public internet alternative; and the red link should be rewarded for its very strong performance over both the public internet and the green link counterfactual scenarios. But this is all incompatible with the carried traffic model. The carried traffic model could be modified with ad-hoc adjustments to distribute rewards for redundancy value, improved latency over the public internet and peers, etc (e.g. insurance lines get paid out at 10% of non-insurance lines). But this general approach would require substantial tuning of its many parameters. Moreover, this approach quickly erodes the simplicity of the carried traffic model anyways.

By contrast, the Shapley value methodology handles these complexities without needing additional components. Shapley values distribute gains of a value function only amongst players, based on each player's marginal contribution to that value function across different coalitions of other players.[1] This avoids the need for managing a wide range of scenarios manually. Contributors only get paid for improvements to the global value function that they specifically drive.

3 Simple Model

There are two key steps to identifying an entity's marginal contribution: defining the aggregate value function and computing its value under different coalitions of network contributors. This section walks through both steps for the model in Figure 2.

The value function starts simple: traffic times the negative latency of the traffic, summed up over all traffic flows. This is a literal manifestation of "Increase Bandwidth Reduce Latency" and represented in Equation (1).

$$V = -\sum_i t_i l_i \tag{1}$$

To compute the value of this function under different coalitions of network contributors, we take the traffic flows as given. Traffic can always reach its destination through the public internet, which has effectively unlimited bandwidth but poor latency. But different coalitions of contributors can improve on the latency on some or all of those flows through their private links. Implicitly, then, DoubleZero only distributes rewards for improvements over the public internet; and not for routes that perform equivalently or worse than the public internet.

In order to compute how an arbitrary coalition of contributors can improve on the value function, we set up and solve a standard linear program that routes traffic and thus delivers a value that the coalition achieves. Critically, this linear program determines only the *hypothetical* routing of traffic, rather than the actual routing of traffic. This is for two reasons. First, all simulated coalitions except for one have only a strict subset of network contributors operating, and so there is no corresponding real-world outcome to measure. Second, for the sole coalition in which all network contributors do operate, the hypothetical and actual routing of traffic may diverge for engineering considerations from time to time. However, the goal of the hypothetical routing logic in this particular scenario is to generally correspond closely to the actual routing logic.

[1] Formally, Shapley values do so by satisfying key desirable properties like symmetry (players with equal contributions get equal rewards), dummy player (those who create no value get no reward), and additivity across both settings and value functions. Indeed, they were a major reason that their creator, Lloyd Shapley, won the Nobel Prize in Economics in 2012.

The linear program has the standard structure, noted in Equation (2), where the vector of decision variables x refers to the amount of traffic to route over each link for each traffic type. In this formulation, c refers to the cost across each link, A_{eq} and b_{eq} ensures that the traffic demands are met, and A_{ub} and b_{ub} ensures that the bandwidth constraints of each link are respected.

$$\min_{x,x\geq 0} cx \quad \text{s.t. } A_{eq}x = b_{eq} \text{ and } A_{ub}x \leq b_{\text{ub}} \tag{2}$$

To illustrate, consider the example in Figure 2, which can be laid out in the following table. For simplicity, the name of the operator is the same color as the link. Suppose we wish to solve the linear program assuming all four operators and their associated four links are made available, i.e. the largest possible coalition of contributors.

Start	End	Latency	Operator
A	B	10	Red
A	B	25	Green
B	C	20	Blue
A	C	25	Orange
A	B	30	Public
B	C	30	Public
A	C	30	Public

Under this formulation, these seven links can be turned into a matrix representing the flow of traffic to and from the various nodes. The matrix has three rows (each representing a node) and fourteen columns (each representing a link, and where the original seven bidirectional links are turned into fourteen directed links). In turn, each entry is a zero if the link is unrelated to the node, and a positive or negative one if it respectively carries traffic away or towards the city. For the reader's convenience, the columns are tagged by the first letter in the link name, and apostrophes represent links facilitating the reverse flow of traffic.

$$\tilde{A}_{eq} = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{cccccccccccccc} r & g & b & o & r' & g' & b' & o' & p_1 & p_2 & p_3 & p'_1 & p'_2 & p'_3 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & -1 & -1 & -1 & 0 & 1 & 1 & 0 & -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & -1 & 1 & 0 & 1 & -1 \end{bmatrix} \end{array}$$

The demand requirements are set against the matrix that embeds the flow logic. This scenario has the following demand matrix, where the row is the origin and the column is the destination.

	A	B	C
A	0	1	0
B	0	0	1
C	1	0	0

However, these three types of traffic represent distinct traffic types and cannot be comingled, as otherwise the outbound traffic from A to B could be offset against the inbound traffic from C to A. Thus, we must represent the demand requirements of each traffic type uniquely. We first focus on the first traffic type, wherein one gigabit of traffic moves from A to B, and generate the following right-hand side expression. In this, the first node A pushes one unit of traffic onto the network, the second node B takes that unit off, and the third node C neither adds nor removes to this particular traffic flow.

$$\tilde{b}_{eq}^1 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

This successfully defines a flow constraint matrix for a single traffic type. From this, we can expand to multiple traffic types by letting x represent not just a single type of traffic being sent over a set of edges, but instead *multiple* types of traffic each being sent over a set of edges independently. This allows us to define a full constraint matrix for all traffic flows in Equation (3).

$$A_{eq} = \begin{bmatrix} \tilde{A}_{eq} & 0 & 0 \\ 0 & \tilde{A}_{eq} & 0 \\ 0 & 0 & \tilde{A}_{eq} \end{bmatrix} \qquad b_{eq} = \begin{bmatrix} \tilde{b}_{eq}^1 \\ \tilde{b}_{eq}^2 \\ \tilde{b}_{eq}^3 \end{bmatrix} \tag{3}$$

The next step in this illustration is to embed bandwidth constraints. These are straightforward. For any particular traffic type, the eight links have their own bandwidth constraints. Public links of course have no constraints.

$$\tilde{A}_{ub} = \begin{array}{c} \begin{matrix} r & g & b & o & r' & g' & b' & o' & p_1 & p_2 & p_3 & p_1' & p_2' & p_3' \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

This scenario has not yet specified the bandwidth limits. For the sake of argument, suppose all links can sustain five gigabits of traffic in either direction, except for the green link, which can only sustain two.

$$b_{ub} = \begin{bmatrix} 5 & 2 & 5 & 5 & 5 & 2 & 5 & 5 \end{bmatrix}^T$$

Together, these define the bandwidth constraint matrix for a single traffic type. As before, we can expand to multiple traffic types by stacking the bandwidth constraints horizontally, as all types of traffic on a single line share that bandwidth. There are no modifications needed for the righthand side constraint.

$$A_{ub} = \begin{bmatrix} \tilde{A}_{ub} & \tilde{A}_{ub} & \tilde{A}_{ub} \end{bmatrix}$$

Finally, the cost matrix in the objective function is simply the vector representation of the latencies for each link.

$$\tilde{c} = \begin{array}{cccccccccccccc} r & g & b & o & r' & g' & b' & o' & p_1 & p_2 & p_3 & p_1' & p_2' & p_3' \\ \end{array}$$
$$\tilde{c} = \begin{bmatrix} 10 & 25 & 20 & 25 & 10 & 25 & 20 & 25 & 30 & 30 & 30 & 30 & 30 & 30 \end{bmatrix}$$

As before, they need to be stacked horizontally to represent the three traffic types in this scenario.

$$c = \begin{bmatrix} \tilde{c} & \tilde{c} & \tilde{c} \end{bmatrix}$$

These five terms of c, A_{eq}, b_{eq}, A_{ub}, and b_{ub} jointly define the linear program. Any standard solver can be used to find the solution. This is a problem where we minimize positive cost, and so we can take the negative value at the optimal point to instead have a term we wish to maximize.

We thus iterate through each of the sixteen possible coalitions – representing the power set of the four operators – and compute out the value function assuming their respective links are present or absent. Figure 3 yields the value function for each of the sixteen combinations. It also indicates the latency that each particular traffic flow must endure, as a helpful intermediate step.

Figure 3: Shapley Value by Coalition

Coalition	ℓ_{AB}	ℓ_{BC}	ℓ_{CA}	Value
No Operators	30	30	30	-90
Red	10	30	30	-70
Green	25	30	30	-85
Blue	30	20	30	-80
Orange	30	30	25	-85
Red, Blue	10	20	30	-60
Red, Orange	10	30	25	-65
Green, Blue	25	20	30	-75
Green, Orange	25	30	25	-80
Blue, Orange	30	20	25	-75
Red, Green	10	30	30	-70
Red, Blue, Orange	10	20	25	-55
Green, Blue, Orange	25	20	25	-70
Red, Green, Blue	10	20	30	-60
Red, Green, Orange	10	30	25	-65
All Operators	10	20	25	-55

The final step is to take this table of coalition-level values and compute operator-level Shapley values. To illustrate, consider a single operator, e.g. the Green operator. In Figure 4, we take all coalitions that *exclude* the green operator, and measure the increase in the value function when the coalition *includes* it. For instance, the Blue and Orange coalition has a value of -75; and this improves to -70 when the Green operator joins the coalition. This improvement of 5 is weighted by the relative number of permutations that correspond to this coalition, assuming every possible ordering of operators.[2]

[2]There are 4! = 24 orderings, and two – Blue, Orange, Green, Red; and Orange, Blue, Green, Red – correspond to this particular scenario; and so the weighting is 2/24.

Figure 4 compares each coalition that includes and excludes the Green operator, along with the marginal improvement and weight. This can be summed to a total Shapley value of 2.5 for the Green operator.

Figure 4: Marginal Value of Green Link

With G	Without G	ΔV	Weight
$V(G,R,B,O)$	$V(R,B,O)$	$(-55)-(-55)=0$	0.250
$V(G,R,B)$	$V(R,B)$	$(-60)-(-60)=0$	0.083
$V(G,R,O)$	$V(R,O)$	$(-65)-(-65)=0$	0.083
$V(G,B,O)$	$V(B,O)$	$(-70)-(-75)=5$	0.083
$V(G,R)$	$V(R)$	$(-70)-(-70)=0$	0.083
$V(G,O)$	$V(O)$	$(-80)-(-85)=5$	0.083
$V(G,B)$	$V(B)$	$(-75)-(-80)=5$	0.083
$V(G)$	$V(\text{none})$	$(-85)-(-90)=5$	0.250
		Sum	**2.50**

The procedure can be repeated for each other network contributor: the high-performance Red operator has a value of 17.5; the semi-performant Blue operator has a value of 10; and the Orange operator, which is only slightly better than the public internet, has a value of 5. In fractional terms, that means that Red operator earns 50% of the available rewards, the Blue operator earns 29%, the Orange operator earns 14%, and the Green operator earns 7%.

4 Adjustments and Complications

The simplified model in Section 3 can be extended to address four complications.

4.1 Operator Withdrawals

Private links can disappear without notice, for commercial, operational, or regulatory reasons. To guard against this, the methodology preemptively rewards redundancy across operators. Formally, it does this by modifying the value function from Equation (1) to Equation (4).

$$V = -\mathbb{E}\left(\sum_i t_i l_i\right) \tag{4}$$

Operationally, this is done by treating the output from the analysis in Figure 3 as the scenario-level values. In other words, this is the value of that coalition *assuming the coalition survives*. By contrast, the **expected** value of the coalition accounts for the fact that certain operators may drop out. That is, suppose a given coalition C is scheduled, then we multiply all subcoalitions within that one times the probability of that particular subcoalition appearing.

$$\mathbb{E}\left(V(C)\right) = \sum_{S \subseteq C} p^{|S|}(1-p)^{|C|-|S|} V(S)$$

In the current model, there is a common operator probability p. In future iterations, this can be made into an operator-specific probability if there is substantial variation between operators.

Computationally, this step can be the primary bottleneck in the methodology if these calculations are done sequentially. Thus, the code recasts the problem to be able to compute the equation for all coalitions in a single pass. Suppose v is the scenario-level values written in cardinal order and B is a diagonal matrix where each element is $p^{|S_k|}$, i.e. the "all-survive" baseline. We then define C to be the Möbius inversion on the Boolean lattice, with the following definition. This can be precomputed to save runtime cycles.

$$C_{ij} = \begin{cases} (-1)^{|S_j|-|S_i|} & \text{if } S_i \subseteq S_j \\ 0 & \text{otherwise} \end{cases}$$

This allows us to compute the expected value for all coalitions in a single operation.

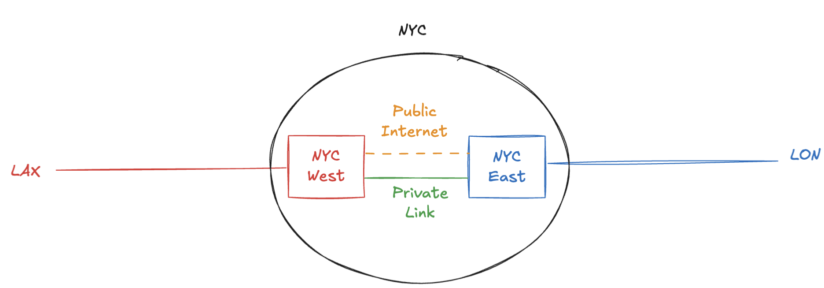
$$\mathbb{E}(V) = CBv$$

4.2 Hybrid Routing

The DoubleZero network can mix public routes with its private links to deliver traffic to its destination. However, such "hybrid routing" is more painful than the latency costs alone would suggest, for multiple reasons. In the short run, the public internet does not give the protocol visibility into bottlenecks, congestion, potential peering disputes, or other problems with traffic flows that could otherwise be proactively solved. In the long run, the public internet does not support the same MTU size that DoubleZero aspires to support, which will introduce traffic limitations or engineering complexities. This suggests that there are negative externalities embedded in hybrid routing that the protocol should try to solve directly.

From an incentives perspective, the solution is to add a penalty to such hybrid routes. In doing so, the protocol introduces an implicit bonus to turning non-contiguous networks into contiguous ones, which is anticipated to be an especially important problem to solve at the boundaries of different contributor links. To illustrate, consider a simple example in Figure 5 wherein two contributors provide links: one from LA to New York, terminating in a switch on the west side of New York; and one from London to New York, terminating in a switch on the east side of New York. The switches are of course connected by the public internet, but a contributor is determining whether to install the green link, that directly connects them privately.

Figure 5: Contiguity Example

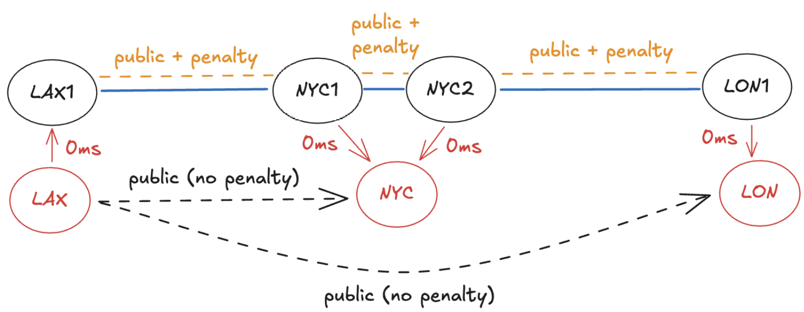


While the rewards methodology formally handles this case, the contributor may be insufficiently motivated to install the green link. The improvements over the public internet latency – which govern the contributor's rewards – may be small over short distances, despite the large positive externalities it unlocks for the network as a whole.

Thus, the rewards model motivates this by applying a penalty to hybrid paths, by adding a latency penalty to the public (orange) link that connects the two switches in New York. This approach delivers a strong incentive for contributors to jointly build contiguous networks. It effectively limits rewards for non-contiguous paths (even if they have low latency) and only contiguous paths will earn the maximum potential rewards. Thus, the contributor who installs the green link internalizes some of that surplus directly.

This can be visualized in Figure 6, and this logic is operationalized in the code. DoubleZero switches are separated from nodes, and nodes have direct public links to one another while switches have mixtures of penalized public links and unpenalized private links between them. As such, the joint private path still needs to beat the public path in order to earn rewards; but the division of those rewards is slightly equalized amongst the different link contributors to encourage contiguity.

Figure 6: Contiguity Example (Programmed Representation)



This approach has some broader benefits. First, by essentially awarding a baseline value to all links regardless of length, it better matches the cost profile of network contributors, who have fixed costs like switches that are similarly incurred regardless of route length. Second, contiguous networks need not only refer to links within the heart of the network: this methodology also encourages reach in the network's periphery, even when those segments may be comparable or even slightly slower than the public internet. Third and above all, this is a market-based solution. This approach, rather than issuing a handful of coarse rules, allows for more flexibility in achieving optimal outcomes. For instance, in certain markets, contiguous connectivity might be outrageously expensive and so a public connection is the better option. The resulting non-contiguous network will not earn rewards at the maximal rate, but it still will be admitted to the network and used ultimately to route traffic.

4.3 Link Jitter

The model in Section 3 assumes that link latencies can be measured as a single number. In practice, there is wide variation around any particular latency measurement, particularly on the public internet. During periods with low congestion, latency might be low; and during periods with high congestion, latency might be high. Moreover, this becomes more complex as the size of the traffic varies. A single packet may face very few queues in its delivery; a stream of packets, though, will see at least some of its members delayed in delivery.

To account for this, we intend on measuring the distribution of link latencies, especially over the public internet, across several sampled time intervals and for a given traffic flow that has meaningful size. We then intend on using some upper percentile value, e.g. the p95, as the "measured" latency for the purposes of the methodology.

4.4 Demand Prioritization and Scale

The model in Section 3 can be extended in two further ways, with respect to demand.

First, not all traffic may be equal. Particularly in a future stage of the network, certain traffic flows may wish to pay for priority access through links with extremely low latency but limited capacity. This can be handled by extended the value function to embed a notion of relative priority p in the traffic flows, taking the value function from Equation (4) to Equation (5).

$$V = -\mathbb{E}\left(\sum_i p_i t_i l_i\right) \tag{5}$$

Second, the rewards methodology distributes rewards for contributors who meet the demand profile as it exists today. However, the procurement for network links can be long, given the substantial capital expenditure needed, and this squares poorly with a fast-growing network. To help grow the capacity of the network in tandem with demand, we propose to scale up current traffic demand by a constant factor γ for routing purposes. Scaling demand helps identify chokepoints or bottlenecks, and distributes rewards in advance to contributors who alleviate those issues. This scaling will evolve as time goes on, and eventually – as the network becomes sufficiently mature – it will be effectively phased out (i.e. set to $\gamma = 1$).

5 Full Model

The full model, with adjustments as described in Section 4, can be found in https://github.com/nihar-dz/network-shapley.

The model can be simulated on a simple network per the script below. These files – corresponding to the map of the private links,[3] the map of public links, and two different demand profiles – are included for user convenience. It is important to note that they do not necessarily represent the current or future map of DoubleZero or anticipated latencies. They are simply cities that are chosen for illustrative purposes, and latencies are computed by finding the distances between their latitude and longitudes and applying some scaling factor with noise for private links and public links alike.

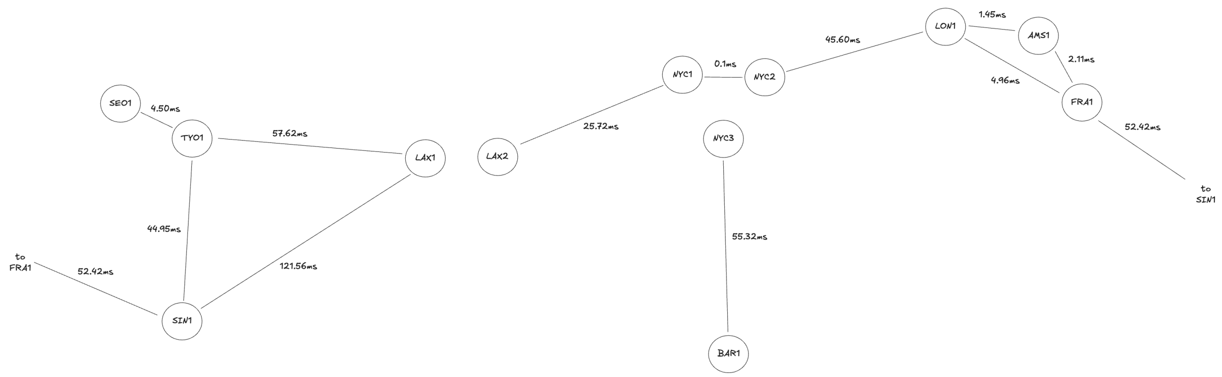
[3]Note that this map currently has a column for a secondary operator of links. This is not an active feature at this time, and so it should be set to NA.

```python
import pandas as pd
from network_shapley import network_shapley

private_links = pd.read_csv("private_links.csv")
public_links  = pd.read_csv("public_links.csv")
demand1       = pd.read_csv("demand1.csv")
demand2       = pd.read_csv("demand2.csv")

result1 = network_shapley(
    private_links     = private_links,
    demand            = demand1,
    public_links      = public_links,
    operator_uptime   = 0.98, # optional
    hybrid_penalty    = 5.0,  # optional
    demand_multiplier = 1.2,  # optional
)
print(result1)

result2 = network_shapley(
    private_links     = private_links,
    demand            = demand2,
    public_links      = public_links,
    operator_uptime   = 0.98, # optional
    hybrid_penalty    = 5.0,  # optional
    demand_multiplier = 1.2,  # optional
)
print(result2)
```

As added context, the map of private links can be visualized in Figure 7. All switches are connected to one another over the public internet of course, but that is not visualized below.

Finally, the function `network_shapley` takes in three additional optional parameters: the probability of any given operator remaining in the network for a given epoch, the latency penalty associated with public links in a hybrid route, and the scaling multiplier for a demand. These correspond to the various complications in Section 4.

Figure 7: Map of Private Links



The results of these simulated scenarios are in Figure 8. In the first simulation, which approximates the Solana map with a leader based in Singapore, Gamma earns most of the rewards, as they provide the valuable Singapore to Frankfurt route that connects the leader to central Europe (where most of the Solana stake lives). Theta, which provides many of the cross-Pacific routes, and Kappa, which provides the routes from Singapore to a meaningful stake presence in Tokyo and a small presence in Seoul, also earn substantial rewards for the same reason.

In the second simulation, there are two concurrent usage patterns: a leader in New York sending blocks to the Solana validator set, and a leader in Buenos Aires broadcasting blocks to several other locations in an equal-weighted manner. In this case, Theta, which again provides cross-Pacific connectivity, and Beta, which provides cross-Atlantic connectivity, earn substantial rewards. In addition, Zeta – which connects New York to Buenos Aires – earns substantial rewards.

Figure 8: Marginal Value of Green Link

Operator Name	**Value #1**	**Percent #1**	**Value #2**	**Percent #2**
Alpha	36.0066	0.0205	27.0097	0.0187
Beta	29.6241	0.0168	298.6752	0.2066
Delta	48.4246	0.0275	160.0689	0.1107
Epsilon	1.4942	0.0008	0.0000	0.0000
Gamma	874.2342	0.4972	71.6711	0.0496
Kappa	241.6948	0.1375	30.9147	0.0214
Theta	526.7842	0.2996	439.1823	0.3038
Zeta	0.0504	0.0000	417.8871	0.2891

Contributors and users of DoubleZero are encouraged to build simulations that are more realistic for their purposes.