The Network Contributor Rewards Model

Nihar Shah

DoubleZero Foundation

May 1, 2025

Outline

Outline

Running the Model in Practice

- Full model in `network_shapley.py`
- Readme, long PDF, and example inputs are in repo
- We'll walk through a minimal working example here

Required Inputs

- **Private links table**: one row per private link (bidirectional)
  `(Start, End, Cost, Bandwidth, Operator1, Operator2 (defunct, set to NA), Uptime, Shared)`
- **Public links table**: one row per private link (bidirectional)
  `(Start, End, Cost)`
- **Demand matrix**: one row per traffic flow
  `(Start, End, Traffic, Type)`

Optional Inputs

- **Operator uptime**: probability that any given operator stays through epoch
- **Hybrid penalty**: latency penalty applied to hybrid private-public routing
- **Demand multiplier**: scalar to adjust demand up or down

Minimal Working Example

```
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")

result = 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(result)
```

Output

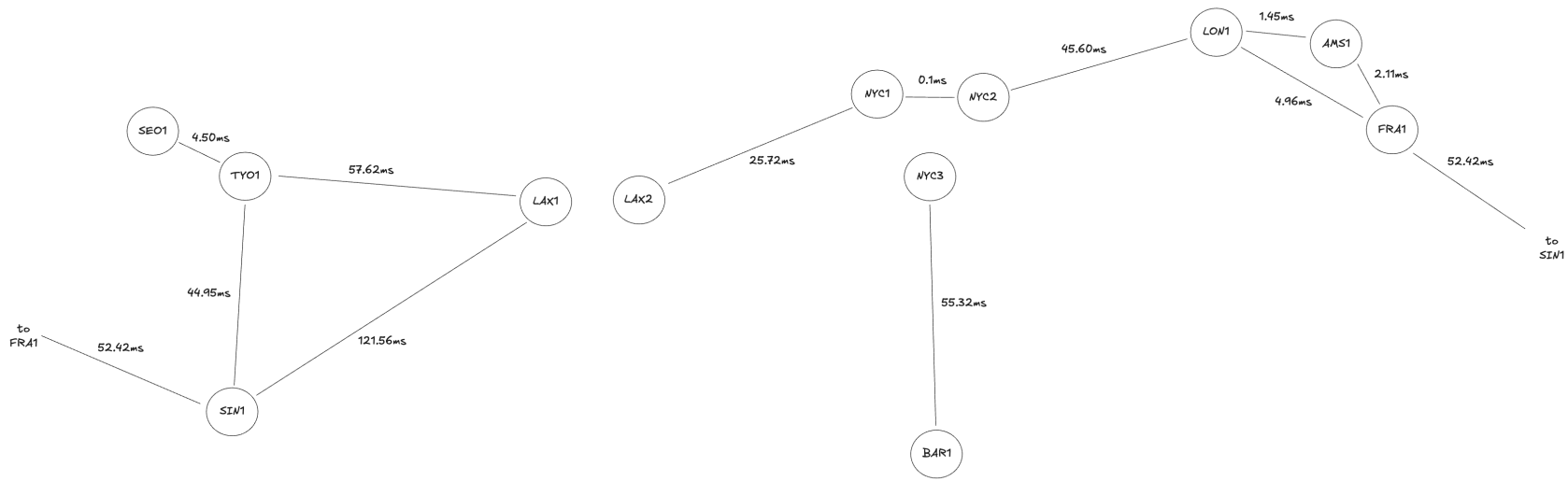
- Returns a `pandas.DataFrame` with columns:

Operator	Name
Value	Raw Shapley value
Percent	$\frac{\text{Value}}{\sum \text{Value}}$

- Links earn rewards based on ability to cut latency and where traffic demand is

- So expect to see large changes depending on where leader is

Simulated Scenario



Simulated Scenario

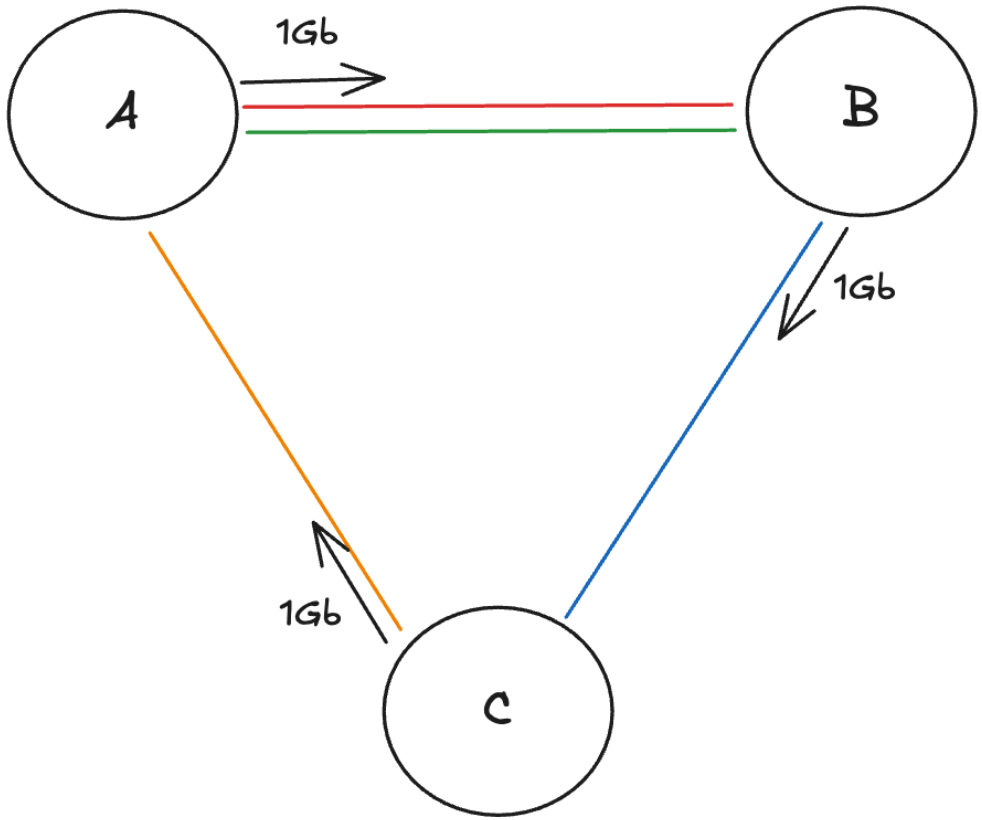
Operator Name	Value #1	Percent #1	Value #2	Percent #2
Alpha	36.0066	0.0205	27.0097	0.0187
Beta	29.6241	0.0168	298.6752	0.2066
Delta	48.4246	0.0275	160.0689	0.1107
Epsilon	1.4942	0.0008	0.0000	0.0000
Gamma	874.2342	0.4972	71.6711	0.0496
Kappa	241.6948	0.1375	30.9147	0.0214
Theta	526.7842	0.2996	439.1823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

Outline

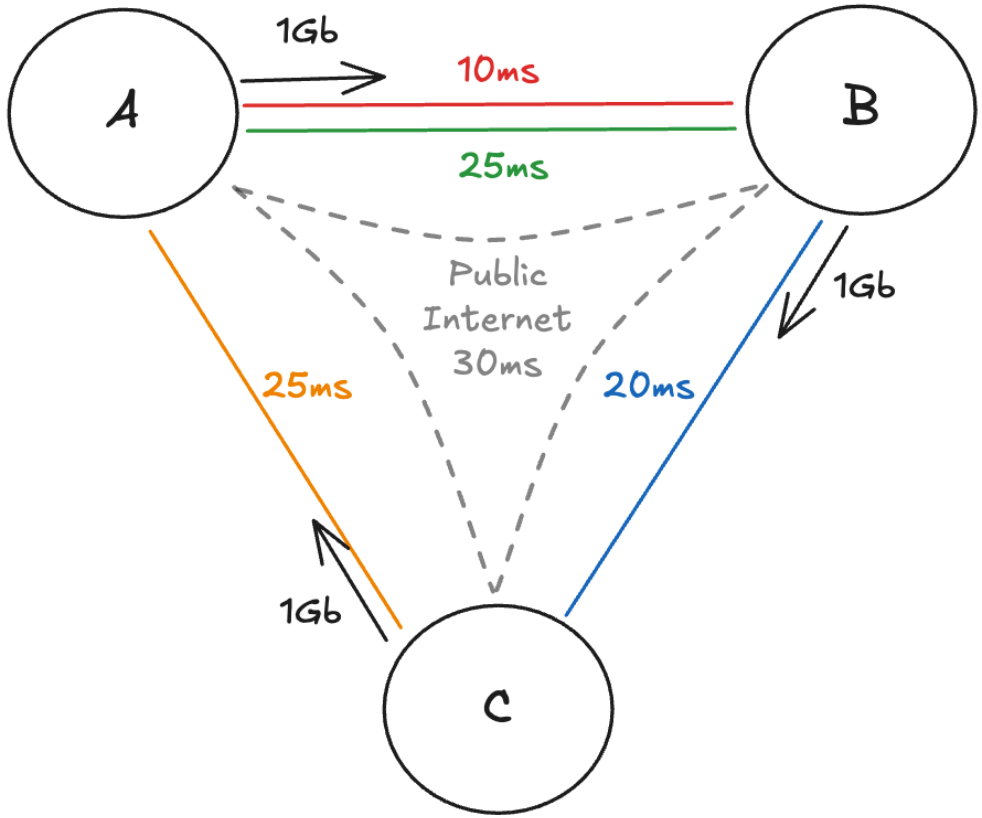
Alternate Idea: Carried Traffic Model

- Carried-traffic model pays only for traffic on each link... seems simple, right?
- But... it is insufficiently discriminating and cannot be kept simple for long
- Shapley values are much more discriminating and robust
- They pay out for the *marginal* contribution to a common **value function**

Illustrative Example



Illustrative Example

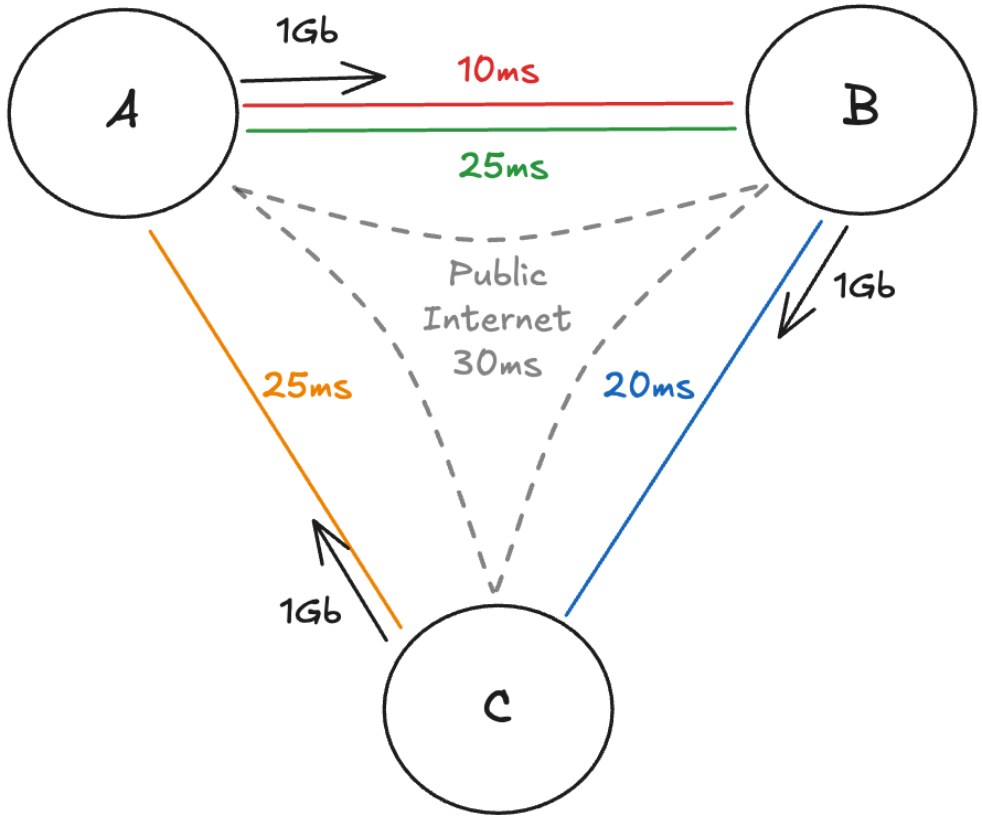


Shapley Values

- Carried-traffic model pays only for traffic on each link... seems simple, right?
- But... it is insufficiently discriminating and cannot be kept simple for long
- Shapley values are much more discriminating and robust
- They pay out for the *marginal* contribution to a common **value function**

Outline

Scenario



Value Function

$$V = -\sum_i t_i l_i$$

- Manifestation of IBRL
- t_i: traffic flow i
- l_i: latency incurred by flow
- Negative sign means we want to maximize this term

Goal: compute this for every coalition of possible contributors

Linear Program Formulation

- To get value for a given set of link operators...

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & A_{eq}\, x = b_{eq}, \\ & A_{ub}\, x \leq b_{ub}, \\ & x \geq 0 \end{aligned}$$

- Decision variables x: traffic routed across each directed edge for each traffic type
- Solving yields coalition-specific value $V(C) = -c^\top x^*$

Private and Public Link Inputs

Start	End	Latency (ms)	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

- Each bidirectional link will be split into two directed edges by the model

Flow-Conservation Matrix

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

- Rows are nodes and columns are directed edges
- $+1$ if traffic leaves the node, -1 if it enters the node
- Need different flow matrices for different traffic types

$$A_{eq} = \text{diag}(\tilde{A}_{eq}, \tilde{A}_{eq}, \tilde{A}_{eq})$$

Demand Vectors

$$\tilde{b}_{eq}^{(1)} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \; \tilde{b}_{eq}^{(2)} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \; \tilde{b}_{eq}^{(3)} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

- Again, need different flow matrices for different traffic types

$$b_{eq} = \begin{bmatrix} \tilde{b}_{eq}^{(1)} \\ \tilde{b}_{eq}^{(2)} \\ \tilde{b}_{eq}^{(3)} \end{bmatrix}$$

Bandwidth Constraints

- Suppose all links but Green have 5 gigabits of capacity; and the Green has 2 gigabits
- Public edges have effectively infinite capacity so there is no constraint imposed on them

$$\tilde{A}_{ub} = [I_{8\times 8}\ 0_{8\times 6}]$$

$$\tilde{b}_{ub} = \begin{bmatrix} 5 & 2 & 5 & 5 & 5 & 2 & 5 & 5 \end{bmatrix}^T$$

- The matrices are stacked horizontally here, because all traffic types draw from the same bandwidth constraint

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

Cost Vector

- Objective minimizes latency, i.e. cost, across traffic types

$$\tilde{c} = \begin{bmatrix} 10 & 25 & 20 & 25 & 10 & 25 & 20 & 25 & 30 & 30 & 30 & 30 & 30 & 30 \end{bmatrix}$$

- Latency is replicated across the three traffic types

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

- This defines all the components needed for the linear program

Value Function Across Coalitions

Coalition	I_{AB}	I_{BC}	I_{CA}	$V(C)$
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

Green Link's Marginal Contribution

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**

- Summing weighted differences yields Green's Shapley value = 2.5
- Red = 17.5 (50%), Blue = 10 (29%), Orange = 5 (14%), Green = 2.5 (7%).

Outline

Operator Redundancy

- Operators may withdraw for commercial, operational, or regulatory reasons
- This methodology would over-reward fragile topologies
- Solution: pay contributors according to the *expected* value of a network and account for probability p of withdrawals
- This preemptively rewards links with insurance value

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

Operator Redundancy

- Formally, turn coalition-specific values into expected values:

$$E\big[V(C)\big] = \sum_{S \subseteq C} p^{|S|}(1-p)^{|C|-|S|}\, V(S)$$

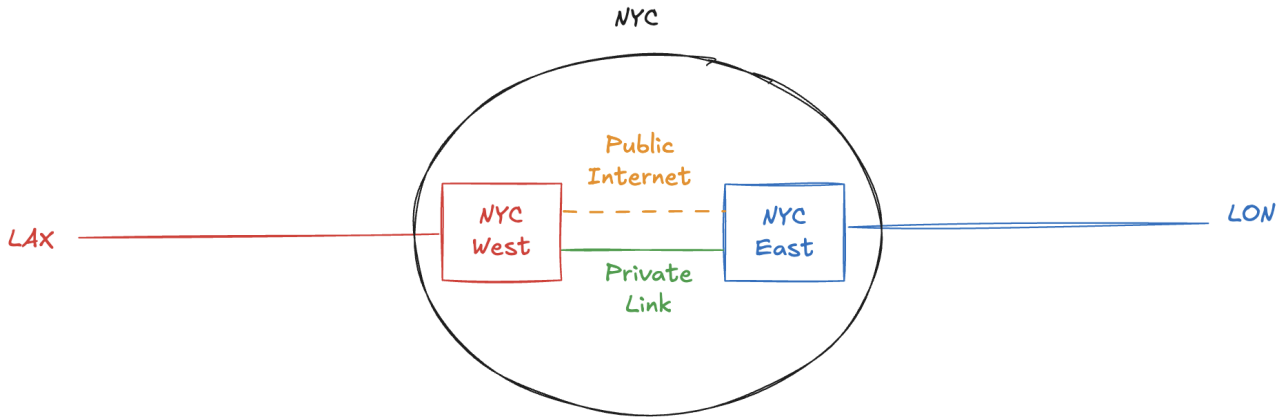
- In practice, this is a computational bottleneck so we use a one-pass algorithm (described in the manual)

$$E(V) = CBv$$

Hybrid Routing and Penalties

- Ideally, DoubleZero would not mix private and public links (visibility risks in short run, MTU limits in long run)
- But contributors might ordinarily build non-contiguous networks
- Solution is to add a latency penalty to every public segment in a mixed path
- This encourages end-to-end deployments, but still admits non-contiguous networks as needed

Hybrid Routing and Penalties



- Orange public link is charged some penalty
- Contributors only unlock the full improvements for LA-London traffic by building the green private link

Measuring Latency

- Real-world latencies are hard to measure as a single number, because they fluctuate with congestion and size
- Solution: measure latency distributions over sampled intervals and non-trivial flow sizes
- Use an upper percentile (e.g. p95) as the effective latency measurement in the value function
- Brings notion of both speed and jitter into rewards calculus without changing core structure

Scaling Demand

- There may be different priorities of traffic; so include this notion in value function

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

- Also, potentially scale current demand by $\gamma > 1$ to anticipate future growth and expose chokepoints
- Gives incentives to invest in these before demand catches up
- Gradually adjust $\gamma \to 1$ as the network matures

Conclusion

- Check out the repo, readme, and manual
- Contact me at nihar@doublezero.us with questions
- On the horizon...
  1. The short-term focus is on measuring quantities and setting parameters accurately
  2. The long-run goal is to extend this to multicast
  3. Ideally we would also design a web interface