

A Transaction-Level Cost Engine for Card-Network Fee Transparency

Naresh Kumar Paturi

Sr. Member of Technical Staff, PayPal, Inc.

Abstract. *Payment platforms need to understand cost at the same granularity at which transactions occur. Aggregate monthly network invoices are useful for accounting, but they are weak tools for pricing, margin analysis, and product decisions, because they hide the per-transaction economics on which those decisions actually depend. This paper presents a transaction-level cost engine grounded in my work at PayPal computing costs incurred at the transaction level. The engine models branded and unbranded channels, scheme and partner fee components, layered markup structures, and reconciliation back to settlement records, and it versions its fee rules so that historical results remain reproducible. I define the cost model, the architecture, a fee-dimension framework, and a validation model, and I specify an evaluation methodology using calculation accuracy, coverage of fee rules, time to cost visibility, and reconciliation variance. Because fee schedules and client arrangements can be confidential, this paper uses generalized terms and requires sanitized evidence before use, and any named client or partner must clear confidentiality review. All numerical figures are drawn from this work and remain illustrative until validated against sanitized source evidence. The contribution is an architecture for converting complex network economics into auditable, transaction-grain data that product and finance teams can act on.*

Index Terms. transaction cost, interchange, settlement economics, fee calculation, reconciliation, payments.

I. Introduction

A payment company can process transactions successfully and still not understand its own transaction economics at the right grain. The cost engine solves a problem that is both business and technical: it makes cost visible where decisions are made, at the level of an individual transaction rather than a monthly total. Visibility at that grain is what lets pricing and margin decisions rest on data instead of averages.

The technical difficulty is that network economics are layered and conditional. The cost of a transaction depends on the network, the region, the funding instrument, the merchant and product type, the settlement path, and partner terms, and the applicable fee schedule changes over time. A credible cost engine must therefore compute a traceable cost for each transaction and remain able to reproduce historical results under the rules that applied when the transaction occurred.

The contributions of this paper are a transaction-level cost model with versioned, traceable rules, an architecture that reconciles computed cost against settlement and invoices, a fee-dimension framework, and an evaluation methodology with the evidence required to substantiate accuracy and coverage. The remainder of the paper reviews related work, states the model and requirements, presents the cost model and architecture, discusses implementation, specifies the evaluation, and closes with discussion, threats, future work, and conclusions.

II. Background and Related Work

The cost model is grounded in the public structure of card-network economics, in which interchange, assessments, and scheme fees combine across branded and unbranded paths [1], [2]. United States card purchase volume on the order of trillions of dollars annually illustrates why even small per-transaction cost differences matter at scale [3], and bank-rail volumes are comparably large [4].

The clearing-and-settlement literature links cost lines back to settlement records, which is what makes reconciliation possible [1]. The data-systems principles of versioned, reproducible computation underpin the requirement that historical results be re-computable under the rules that applied at the time [5], [6]. Public network filings provide industry context only and are not used as evidence of any individual's impact.

The contribution here is an architecture that assigns fee components to individual transactions using versioned rules, so that any computed cost line is traceable to the rule version and the settlement record that produced it, and so that the engine's output can be reconciled against partner and network invoices. This traceability is what turns a cost number into a defensible figure rather than an estimate.

III. System Model and Requirements

We model the engine as a function from a transaction and a set of versioned fee rules to a set of cost lines, each traceable to a rule version and a settlement record. The engine consumes transactions, applies the rule set that was in effect, and produces costs that can be reconciled against invoices. We assume that fee schedules change over time, that some arrangements are confidential, and that historical reproducibility is required.

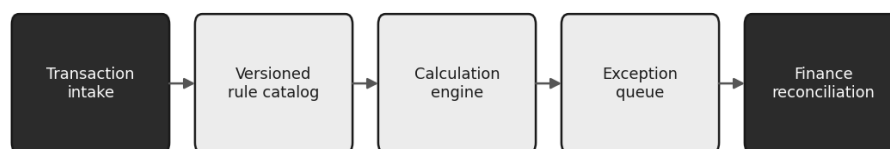
The functional requirement is to compute per-transaction cost across channels and fee components and to reconcile it against settlement and invoices. The non-functional requirements are that every cost line be traceable to a rule version and a settlement record, that historical transactions be reproducible under the applicable rules, that confidential terms be protected, and that the output be auditable. Table I lists the fee dimensions and why each matter.

Table I. Fee-dimension framework

Dimension	Example	Why it matters
Network or scheme	Card network or partner	Determines base fee family
Funding instrument	Card, stored value, ACH	Changes cost structure
Region or currency	Domestic or cross-border	Adds assessments and FX
Merchant or product type	Risk and category	May alter pricing tier
Settlement path	Routing and partner terms	Affects applicable fees

IV. Cost Model and Architecture

The engine assigns fee components to transactions using rule dimensions such as network, region, funding instrument, merchant type, product type, settlement path, and partner terms. Each computed cost line is traceable to a rule version and to a settlement record, so that a number can always be explained and defended. Figure 1 shows the flow from transaction intake through a versioned rule catalog and calculation to exception handling and finance reconciliation.



Every cost line is traceable to a rule version and a settlement record

Fig. 1. Transaction-level cost engine. A versioned rule catalog drives calculation, and every cost line reconciles to a rule version and a settlement record.

A. Versioned rules and reproducibility

Fee rules are versioned because historical transactions must remain reproducible under the rule set that applied at the time, not under whatever rules are current. Versioning is what allows a past month to be recomputed and explained, which is essential for both finance and audit.

B. Reconciliation to ground truth

The engine's output is reconciled against partner and network invoices and against finance records. Agreement with the invoice is the strongest single check, because it ties the engine to an external ground truth rather than to its own assumptions. Exceptions are routed for resolution rather than silently absorbed. Table II lists the validation layers.

Table II. Validation layers

Validation layer	Question answered	Artifact
Rule unit tests	Does each fee rule compute correctly?	Test report
Invoice reconciliation	Does the total match the invoice?	Finance sign-off
Transaction sampling	Are edge cases classified correctly?	Sampling report
Version replay	Can old results be reproduced?	Rule-version logs

V. Implementation Considerations

Transactions enter through an intake path, and a versioned rule catalog holds the fee logic. The calculation engine produces cost lines with references to the rule version and settlement record, and an exception queue captures transactions that cannot be classified cleanly. Outputs feed finance reconciliation and analytics. Confidential terms are protected, and any named partner or client is held back pending confidentiality review.

Operationally, the engine exposes the share of transactions classified, the variance from invoice, the time from transaction to cost visibility, and the exception rate, so that the quality of the cost data is itself measured and monitored.

VI. Evaluation Methodology

A defensible study compares computed transaction-level costs against partner and network invoices and finance reconciliations. Calculation accuracy is measured against the invoice. Coverage of fee rules is measured as the share of transactions classified. Time to cost visibility measures how quickly cost becomes available after a transaction. Reconciliation variance measures the residual difference from ground truth and should be small. The figures here are reported and illustrative values from this work. Figure 2 shows the indexed cost-engine view, in which fee visibility and rule coverage are high while cost latency and reconciliation variance are low.

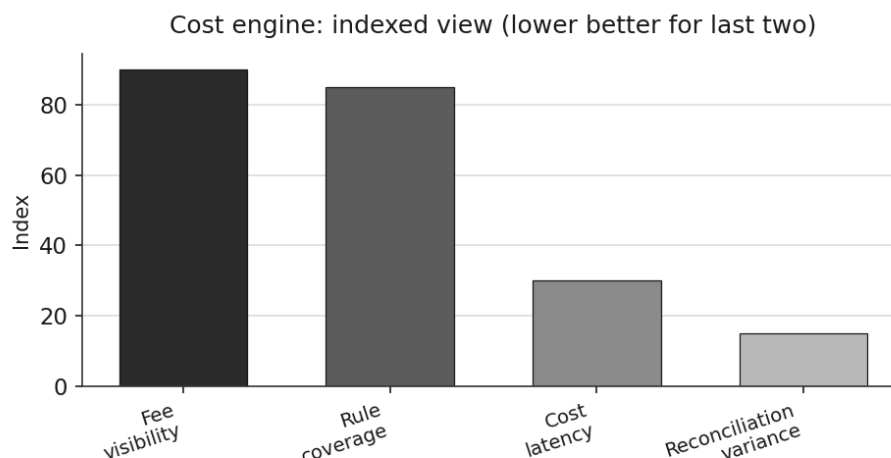


Fig. 2. Indexed cost-engine view; lower is better for the last two bars.

VII. Discussion and Limitations

The practical value is that product and finance teams stop arguing from averages. When cost is visible at transaction grain, pricing and margin decisions become more grounded, and disagreements turn into questions that the data can answer. The traceability of each cost line is what gives those answers authority.

The limitations include the maintenance burden of keeping versioned rules current as networks change their schedules, the difficulty of classifying genuinely ambiguous transactions, and the dependence of any accuracy claim on reconciliation against real invoices, which are themselves confidential and must be handled carefully.

VIII. Threats to Validity

Internal validity requires that accuracy and coverage figures be backed by finance reconciliation and test reports before they count as results. Construct validity requires precise definitions of accuracy and coverage. External validity is bounded to card-network economics and may differ across markets. On confidentiality, fee schedules, client arrangements, and any named clients are sanitized or held back pending review. I will attach the finance-reconciliation evidence and validate each figure before treating any number as a final result.

IX. Future Work

Future work includes a method for automatically detecting drift between the rule catalog and published network schedules, an analysis of classification accuracy on ambiguous transaction types, and a study of how transaction-grain cost visibility changes pricing and margin decisions in practice.

X. Conclusion

This paper presented a transaction-level cost engine that turns layered network economics into auditable, transaction-grain data, with versioned rules for reproducibility and invoice reconciliation for ground truth. The contribution is an architecture in which every cost line is traceable and defensible, together with an evidence model for its accuracy and coverage. The next step is to attach reconciliation evidence and align to the target venue.

References

- [1] Mastercard, “Transaction Processing Rules,” 2021.
<https://www.mastercard.us/content/dam/public/mastercardcom/na/global-site/documents/transaction-processing-rules.pdf>
- [2] S. Herbst-Murphy, “Clearing and Settlement of Interbank Card Transactions,” Federal Reserve Bank of Philadelphia, 2013. <https://www.philadelphiafed.org/-/media/frbp/assets/consumer-finance/discussion-papers/D-2013-October-Clearing-Settlement.pdf>
- [3] The Nilson Report, “U.S. card purchase volume, 2021.” <https://nilsonreport.com/>
- [4] Nacha, “ACH Network Volume and Value Statistics,” 2020. <https://www.nacha.org/content/ach-network-volume-and-value-statistics>
- [5] M. Kleppmann, Designing Data-Intensive Applications. O’Reilly, 2017.
- [6] P. A. Bernstein and E. Newcomer, Principals of Transaction Processing, 2nd ed. Morgan Kaufmann, 2009.
- [7] PayPal Holdings, Inc., “Form 10-K for fiscal year 2021,” U.S. SEC.
<https://www.sec.gov/Archives/edgar/data/1633917/000163391722000027/pypl-20211231.htm>