TORC3: Token-Ring Clearing Heuristic for Currency Circulation

Carlos Humes Jr.[*], Marcelo S. Lauretto[†], Fábio Nakano[†], Carlos A.B. Pereira[*], Guilherme F.G. Rafare[**] and Julio M. Stern[*,‡]

[*]*Institute of Mathematics and Statistics of the University of São Paulo, Brazil*
[†]*School of Arts, Sciences and Humanities of the University of São Paulo, Brazil*
[**]*FinanTech Ltda.*
[‡]jmstern@ime.usp.br

Abstract. Clearing algorithms are at the core of modern payment systems, facilitating the settling of multilateral credit messages with (near) minimum transfers of currency. Traditional clearing procedures use batch processing based on *MILP* - mixed-integer linear programming algorithms. The *MILP* approach demands intensive computational resources; moreover, it is also vulnerable to operational risks generated by possible defaults during the inter-batch period. This paper presents *TORC3* - the Token-Ring Clearing Algorithm for Currency Circulation. In contrast to the *MILP* approach, *TORC3* is a real time heuristic procedure, demanding modest computational resources, and able to completely shield the clearing operation against the participating agents' risk of default.

Keywords: Bayesian calibration; Clearing algorithms; Heuristic optimization; Multilateral netting; Operational risk; Payment systems; Real-time heuristic procedures.

1. INTRODUCTION

Clearing algorithms are at the core of modern payment systems, facilitating multilateral netting, that is, the settling of multilateral credit messages with (near) minimum transfers of currency. Traditional clearing procedures use batch processing based on mixed-integer linear programming algorithms. Large scale mixed-integer programming demands intensive computational resources; most of the traditional clearing algorithms require large mainframe computers. Furthermore, this approach delays all payments to after the batch processing (usually overnight). Moreover, batch processing introduces an operational risk originated from the possibility of a participant's default in the iter-batch processing period.

This paper presents *TORC3*, the Token-Ring Clearing Algorithm for Currency Circulation, a real time heuristic procedure. In contrast to batch mixed-integer programming procedures, *TORC3* demands only modest computational resources. Furthermore, *TORC3* is able to completely shield the clearing operation against the participating agents' risk of default.

TORC3 was conceived as the core algorithm for a clearing system to be used at *CIP-SBP* the Interbank Payments Clearing House of the Brazilian Payment System (Câmara Interbancária de Pagamentos do Sistema Brasileiro de Pagamentos). *TORC3* handles only the algorithmic aspects of the clearing process. Its spartan implementation in plain *ANSI C* language has less than ten thousand lines of code, see Rafare et al.

XI Brazilian Meeting on Bayesian Statistics
AIP Conf. Proc. 1490, 179-188 (2012); doi: 10.1063/1.4759602

(2001). A host of additional tasks related to the pre-processing of individual credit messages, communication and cryptography, process traceability, accounting reports and on-line auditing, system security, etc. are handled by other sub-systems. However, it was always a fundamental requirement of system design to maintain these sub-systems as structurally independent as possible, following the best practices of software engineering. The present paper presents only the core *TORC3* algorithm; future articles will present other sub-systems and extensions of the basic heuristic for a variety of applications.

Section 2 gives a high level description of *TORC3* Algorithm. Section 3 presents some simulation results, makes brief comments about pre-processing, suggests directions for further research and gives our final remarks.

2. THE *TORC3* CLEARING ALGORITHM

Clearing systems handle multilateral credit messages between participating agents. For example, at an interbank clearing system all participating agents are state, commercial or investment banks. The clearing system receives credit messages and tries to clear them, that is, to execute them all, by netting, that is, by mutual multilateral cancellation, using actual transfers of currency only as a last resort.

A credit message M stipulates a payment, in the amount of v standard units, to be made by agent-i in favor of or to be received by agent-j. The information pertinent to a credit message, M, is specified in a list, $[t, i, j, v]$, having the following fields:

t - Time stamp and identification number;
i - Origin;
j - Destination; and
v - Value, specified as an integer number of standard units.

In order to protect the clearing system, shielding it against risks of default, each participating agent must guarantee all its operations. This assurance is provided by three reserves maintained at the clearing system by each agent:

$R1$ - Monetary reserve. The monetary reserve represents capital in the form of currency, a resource that cannot be invested or used in any form outside the clearing system. Hence $R1$ represents a cost or loss of opportunity to the agent. Therefore, the clearing system is designed to keep it at very low level, that is, $R1$ should correspond to a small fraction of the agent's daily operations.

$R2$ - Collateral reserve. The collateral reserve consists of financial assets kept in custody of the clearing system.

$R3$ - Ticket reserve. The ticket reserve is the collection of tickets held by a participant at a given time. A ticket is an IOU (I-owe-you), an acknowledgement of debt sent by another agent. Tickets and tokens are short-living objects that are strictly internal to *TORC3*, as explained in the sequel.

When agent-i, sends a credit message for clearing, the system makes sure that agent-i's reserves can back it up. The following condition checks this situation, as it will

become clear from further details of the clearing process.

$$\text{Condition I:} \quad R1(i) + \min[R3(i), R2(i)] \geq v \ ,$$

The system also prevents excessive concentration of monetary reserve at the credit message destination, j, a situation that could result in *TORC3* stalling or loss of efficiency. This is accomplished by checking if the execution of this credit message would make $R1(j)$ exceed a stipulated maximum,

$$\text{Condition II:} \quad R1(j) + v \leq \text{MaxR1}(j) \ .$$

If conditions I and II are satisfied, the credit message is accepted for clearing; otherwise, it is rejected. Rejected messages are handled by external sub-systems that can make the payment directly or resubmit the credit message to the clearing system at a later time.

The first step in processing a message is its disassembling into unitary (value-1) transactions. Each v-valued message from agent-i to agent-j is disassembled into v unitary transactions from i to j.

The first basic idea used to develop *TORC3* is to find sets of unitary transactions that can be matched in closed cycles. The transactions in such a cycle cancel each other, having a null resultant, that is, they can be settled with no actual transfer of currency.

The second basic idea is to provide a "buffer" that stores these transactions for a limited time. During this time, a stochastic optimization heuristic tries to reduce "noise" or fluctuations in the incoming messages by cancellation of cyclic transaction sets.

When a transaction enters the buffer, a ticket is sent from agent-i to agent-j.

Case 1: If $R3(i) > 0$ and $R2(i) > 0$ then agent-i forwards a ticket from its reserve to agent-j, $R3(i)$ is decremented and $R3(j)$ is incremented. Since a ticket is just an IOU from a third party, agent-i must back up this operation decrementing its collateral reserve, $R2(i)$. We represent these operations using the synthetic notation: $R3(i)--$, $R2(i)--$, $R3(j)++$.

Case 2: Else, If $R1(i) > 0$, agent-i creates a ticket and sends it to agent-j. Creating a ticket implies decrementing $R1(i)$ and incrementing $R3(i)$, so the process is represented as:
$R1(i)--$, $R3(i)++$, $R3(i)--$, $R3(j)++$, what is equivalent to
$R1(i)--$, $R3(j)++$.

Case 3: Else, there is an inconsistency with Condition I, that is, $R1(i) + \min[R3(i), R2(i)] < v$, and the original credit message should have been rejected.

A ticket carries information concerning its time of creation, t, and its path, listing the agents that have owned it. The ticket's path, $[a, b, \ldots x]$, starts with its creator, a - at the head of the list, and ends with its current owner, x - at the tail of the list. Notice that, according to the rules defined above to create and forward tickets, the operation at the head of a ticket is guaranteed by $R1$, monetary reserves used in Case 1, while the following operations are backed up by $R2$, collateral reserves used in Case 2.

A circuit in the ticket's path corresponds to a zero-resultant financial exchange. *TORC3* accomplishes multilateral nettings by the elimination of circuits in tickets, and the elimination of the corresponding transactions. When a circuit covers the whole

ticket's path, we have a cycle. Hence, when a cycle is eliminated, the whole ticket is eliminated.

At one hand, the longer tickets are allowed to exist in the buffer, the longer will be their trajectories, and the better the chances for circuit or cycle formation and elimination. At the other hand, existing tickets deplete the participating agents' reserves; hence, tickets should not be allowed to accumulate for too long in the buffer. Therefore, in order to obtain a good performance from the *TORC3* heuristic, the clearing system operator must set and fine tune `MaxTicketLife` - the tickets' maximum permitted life-span. When a ticket reaches the maximum life-span, it is eliminated. Hence, also tickets with non-cyclic (lists of) transactions will be generated and eliminated by *TORC3* in the clearing process. The rules for ticket and circuit elimination are explained in the next sub-section.

Let us briefly discuss possible criteria for choosing which ticket to forward in a transaction, when more than one is available, i.e. if $R3(i) \geq 2$. The following deterministic heuristic combines three simple criteria:

1. If possible, forward a ticket that creates a cycle; if more than one is available, choose the longest cycle.
2. Else, if possible, forward a ticket that creates a circuit; if more than one is available, choose the longest circuit.
3. Else, in the last case, forward the oldest available ticket.

Alternative, and more efficient, randomized heuristics choose among the available tickets according to probabilities that are proportional to a scoring function taking into account the three simple criteria of the deterministic heuristic.

2.1. Tickets, Tokens and Circuit Elimination

TORC3 can eliminate an entire ticket with no circuits, or eliminate a ticket's sub-list of transactions forming a circuit, or eliminate a cyclic ticket. The transactions in an eliminated list are transformed into tokens, that are released from the buffer. A token can be of one of five types, according to the position of the corresponding transaction in the original ticket. The five token types are:

A - Initial in a path,	$a \Rightarrow b \rightarrow c \rightarrow d$;
E - Intermediate,	$a \rightarrow b \Rightarrow c \rightarrow d$;
I - Terminal in a path,	$a \rightarrow b \rightarrow c \Rightarrow d$;
O - Initial in a cycle,	$a \Rightarrow b \rightarrow c \rightarrow a$;
U - Singular or unit-length path,	$a \Rightarrow b$.

The information pertinent to a token, T, is specified in a list, $[i, j, k, l]$, having the following fields:

i - Origin;
j - Destination;
k - Type;
l - Value.

TABLE 1. Reserve operation for each ticket transaction and token destruction.

Type	Ticket transaction	Token destruction	Message rejection
A	$R1(i)--$	nop	$R1(i)++$
E	$R2(i)--$	$R2(i)++$	$R2(i)++$
I	$R2(i)--$	$R2(i)++$ and $R1(j)++$	$R2(i)++$
O	$R1(i)--$	$R1(i)++$	$R1(i)++$
U	$R1(i)--$	$R1(j)++$	$R1(i)++$

For the sake of simplicity we consider in this article only tickets and tokens of unitary value. However, *TORC3* can be easily generalized to handle tickets in a convenient range of pre-selected values, like $l \in \{1,2,5,10,20,50,100\}$; or $l \in \{1,3,10,30,100\}$.

The tokens with destination j released by the buffer go to agent-j's assembly line, where the credit messages with the same destination wait in a queue to be reassembled. Notice that a token arriving to agents-j's assembly line is used to reassemble the first message in line, independent of its origin.

When a message is reassembled, it is released by *TORC3* sub-system, and passed to external sub-systems that shall handle actual financial operations, accounting reports, etc. Used tokens are destroyed as the message is released.

When a token is destroyed, reserves used to back-up corresponding ticket transactions are released. Table 1 summarizes the rules used to increment and decrement the pertinent reserves at the time of a ticket transaction and at the time of the corresponding token destruction. The destruction of tokens of type E and O, just restore the reserves used at the corresponding ticket transaction; these operations cancel each other, having a null resultant. The destruction of a pair of tokens, of type A and I, at the head and the tail of a tickets (non-cyclic) path, $[a,b,\ldots x]$, have the net effect of transferring one unit of monetary reserve from the agent at the path's head, a, to the agent at the path's tail, x. The token of type U handles the special case of a singular or unit-length path.

Finally we give some illustrative examples for the elimination process. Arrows represent ticket transactions. Reserve decrements at ticket transaction time are annotated above each arrow. Corresponding token types and reserve increments at the time of token destruction are annotated under the arrows.

Path:

$$\begin{array}{ccccccc} & R1(a)-- & & R2(b)-- & & R2(c)-- & \\ a & \Longrightarrow & b & \Longrightarrow & c & \Longrightarrow & d \\ & A & & E & & I & \\ & & & R2(b)++ & & R2(c)++,\ R1(d)++ & \end{array}$$

Circuit:

$$\begin{array}{ccccccc} & R1(a)-- & & R2(b)-- & & R2(c)-- & \\ a & \Longrightarrow & b & \Longrightarrow & c & \Longrightarrow & b \\ a & \Longrightarrow & b & E & & E & \\ & & & R2(b)++ & & R2(c)++ & \end{array}$$

Cycle:

$$a \;\; \begin{array}{c} R1(a)-- \\ \Longrightarrow \\ O \\ R1(a)++ \end{array} \;\; b \;\; \begin{array}{c} R2(b)-- \\ \Longrightarrow \\ E \\ R2(b)++ \end{array} \;\; a$$

Singular or unit-length Path:

$$a \;\; \begin{array}{c} R1(a)-- \\ \Longrightarrow \\ U \\ R1(b)++ \end{array} \;\; b$$

The "real time" operational characteristics of the clearing system require that credit messages can only wait a limited time to be executed. If a credit messages time in *TORC3* sub-system exceeds `MaxMessageLife` the order is rejected. Rejected messages are handled by external sub-systems that can make the payment directly or resubmit the credit message to the clearing system at a later time. Table 1 summarizes the rules used to increment and decrement the pertinent reserves for tokens destroyed when a partially assembled message is rejected. This kind of message rejection should not occur if the clearing system operates under coherent conditions, including `MaxTicketLife` $<$ `MaxMessageLife`. Hence, the last column of Table 1 is only important for exception handling and error recovery situations.

3. SIMULATIONS AND PERFORMANCE

FinanTech and *Banco do Brasil* have established three benchmark scenarios used to test a *TORC3* prototype. These scenarios were built according to statistical profiles typical of the 1999-2000 operations at *CIP-SBP*, the Clearing House of the Brazilian Interbank Payment System. Scenarios 1, 2 and 3 are presented in Table 2 and have, respectively, 30, 100 and 173 participants. Participating agents are ranked according to their size, measured by typical transaction volume for clearing. Table 2 presents the very realistic Scenario 3 in condensed form, grouping together small participants of comparable size. In each of the tree scenarios, a stream of 300,000 random credit messages is generated according to the following specifications: The value of each message is distributed uniformly between 1 and 100 standard units. The origin and destination of each message are distributed with weights proportional to each of the participating agent's size. These random streams of credit messages where given as input to the *TORC3* prototype over a period of 480 minutes, representing the standard 8 daily business hours of commercial banks operating in Brazil.

The *TORC3* prototype is coded in plain *ANSI-C*, compiled usig *GCC*, and implemented in a *Intel Pentium III* 700 MHz computer running *Linux*. At the start of each simulation, an auxiliary Bayesian calibration procedure allocates the monetary reserves required for each participant relative to its expected transaction volume.

Table 3 gives the minimum (for the larger agents), maximum (for the smaller agents) and average required monetary reserve, as a percentage of the agent's expected transaction volume. The collateral reserves are always assumed to be sufficient. Hence, in these

TABLE 2. Simulation Scenarios

Group	Agents	Agent Size %	Group Size %	Accumulated %
Simulation Scenario 1 (30 Agents)				
1	1	23	23	23
2	1	17	17	40
3	1	15	15	55
4	1	13	13	68
5	2	10	20	88
6	24	0.5	12	100
Simulation Scenario 2 (100 Agents)				
1	4	15	60	60
2	6	5	30	90
3	10	0.2	2	92
4	80	0.1	8	100
Simulation Scenario 3 (173 Agents)				
1	1	18.20	18.20	18.20
2	1	14.49	14.49	32.69
3	1	14.20	14.20	46.89
4	1	8.27	8.27	55.16
5	1	6.58	6.58	61.74
6	1	4.64	4.64	66.38
7	1	3.80	3.80	70.18
8	1	3.59	3.59	73.77
9	1	3.47	3.47	77.24
10	1	2.70	2.70	79.94
11	1	2.14	2.14	82.08
12	1	1.74	1.74	83.82
13	1	1.26	1.26	85.08
14	1	1.22	1.22	86.30
15	1	0.94	0.94	87.24
16	2	0.58	1.16	88.40
20	1	0.30	0.30	88.70
21	25	0.19	4.75	93.45
23	20	0.15	3.00	96.45
24	25	0.08	2.00	98.45
25	35	0.03	1.05	99.50
27	50	0.01	0.50	100.00

simulations, rejections may occur by lack of $R1$, never by insufficient $R2$. From the CPU idle time shown in Table 3, one can see that, even using very modest computational resources, the *TORC3* prototype could easily handle the task at hand. Moreover, the message rejection rate was below 3%, with a total amount of monetary reserves required from the participating agents below 0.4% of the total transacted value.

FinanTech and *Banco do Brasil* considered these statistics as indicators of excellent performance. Nevertheless, as expected, the simulations show that the required monetary

TABLE 3. Simulation Results

Data	Scenario 1	Scenario 2	Scenario 3
Agents	30	100	173
Cred.Messages	300,000	300,000	300,000
Accepted Messages (% number)	97.0	95.8	95.7
Accepted Messages (% volume)	97.8	96.9	96.1
Average Message size	50	50	50
Max. Message Life (minutes)	15	15	15
Simulated clearing period (minutes)	480	480	480
CPU idle time (%)	95	81	58
$R1$ (% of agent's volume), min	0.3	0.3	0.3
$R1$ (% of agent's volume), mean	0.38	1.15	5.1
$R1$ (% of agent's volume), max	0.4	1.3	13.0

reserves must be relatively higher for small participants. This observation suggests the possibility of organizing cooperating groups of small agents in order to decrease their operating costs in the clearing process.

4. FINAL REMARKS

Pre-Processing

The standard unit of value for credit messages used by *TORC3* has to be calibrated for optimal performance. For the test problems at hand, the standard unit was set to make the average credit message of size 50. This standard unit is much larger than typical payment orders generated by individual costumers of *CIP-SBP* banks, like checks, bills, credit card statements, etc. Therefore, a pre-processing algorithm is used to bind many small payments into larger credit orders between agents of the clearing system. This pre-processing algorithm is based on approximate solutions for the knapsack-packing problem described in Lau (1986, Ch.3).

Each credit message incorporates the additional field, `change`, used to specify the difference between the total value of all real payments bound together in a credit order treated by clearing system and that order's value specified as an integer number of standard units. When the credit order is completed and released by *TORC3*, the corresponding `change` is sent back from the monetary reserve (R1) of the agent at the credit message destination to the agent at its origin.

Conclusions

We presented *TORC3* - The Token-Ring Clearing Heuristic for Currencies Circulation. In contrast to traditional clearing algorithms based on *MILP* - mixed-integer linear

programming algorithms, *TORC3* presents the following advantages (at least under the simulated scenarios presented in section 3):

- *TORC3* offers continuous-time processing, versus *MILP*'s discrete batch approach.
- *TORC3* demands a maximum payment delay of 15 minutes, versus *MILP*'s post batch processing payment (typically overnight).
- *TORC3* offers complete operational shielding from the participating agents' risk of default, versus possible inter-batch exposures.
- *TORC3* achieves a high rate of netting, over 95%, requiring only modest monetary reserves, bellow 0.4% of transacted volume. This performance surpasses well established industry standards based on the reported efficiency of *MILP*-based commercial clearing systems.
- *TORC3* requires only modest computational resources, in contrast to mainframe-based commercial clearing systems.

Future Developments

We plan to study possible uses of clearing heuristics beyond their most traditional range of applications. We are especially interested in small and medium scale applications, including private clearings that can be operated by indirect credit providers. For example, private clearings could be successfully employed by credit, debit or service card providers, consortia of small business, or cooperative enterprises. With these goals in mind, the following items are presently under development:

R1 Pilot - A dynamic Bayesian calibration procedure that replaces the static monetary reserve allocation procedure mentioned at section 3.

TORC4 - The Token-Ring Clearing Heuristic for Complementary Currencies Circulation. It extends the scope of *TORC3* by permitting the use of multiple circulating currencies under complex and dynamic conversion rules.

Acknowledgements

The authors are grateful for the support of *Politec Tecnologia da Informação S.A.* and *Banco do Brasil*, our partners in the *CIP-SBP* development project. The authors are also grateful to Profs. Jack Baczynski and Pedro Leite da Silva Dias for having invited this paper for the Minisymposium on Stochastic Modeling Applied to Finance to be helt at the 5th LNCC Meeting on Computational Modeling, an initiative of LNCC - the National Laboratory for Scientific Computing, in Petrópolis, Brazil. Finally, the authors are grateful for the support of USP - the University of São Paulo, CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico, and FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo, including program CEPID - Centros de Pesquisa, Inovação e Difusão.

REFERENCES

Deban (2009). *Brazilian Payment System.* Deban - Department of Banking Operations and Payments System, Central Bank of Brazil.

M.J.Fry, I.Kilato, S. Roger, K.Senderowicz, D.Sheppard, F.Solis, J.Trundle (1999). *Payment Systems in Global Perspective.* London: Routledge.

M.Güntzer, D.Jungnickel, M.Leclerc (1998). Efficient Algorithms for the Clearing of Interbank Payments. *European Journal of Operational Research,* 106,1, 212-219.

A.G.Haldane, S.Millard, V.Saporta (2008). *The Future of Payment Systems.* London: Routledge.

H.T.Lau (1986). *Combinatorial Heuristic Algorithms with FORTRAN.* NY: Springer.

R.A.S.Peñaloza (2011). Implementation of Optimal Settlement Functions in Real-Time Gross Settlement Systems. Tech.Rep. 353, Dept.of Economy, University of Brasília.

R.A.S.Peñaloza (2009). A Duality Theory of Payment Systems. *Journal of Mathematical Economics,* 45, 9-10, 679-692.

G.F.G.Rafare, J.M.Stern, C.A.B.Pereira, F.Nakano (2001). Algoritmo de Concretização e Compensação de Mensagens de Pagamento. Tipo de Programa: AT03, AT04, TC02; Campos de Aplicação: EC04, EC06, FN03, FN04. Número de registro no INPI: 00042036 (patent number at the Brazilian Institute of Industrial Property).

D.Rambure, A.Nacamuli (2008). *Payment Systems: From the Salt Mines to the Board Room.* Basingstoke, Hampshire: Palgrave Macmillan.