State-Driven Workload Generation in Distributed Systems: System Model of an FDPS

Roberto Natella[1] and Fabio Scippacercola[2]

[1] Università degli Studi di Napoli Federico II
roberto.natella@unina.it
[2] Consorzio Interuniversitario Nazionale per l'Informatica
fabio.scippacercola@consorzio-cini.it

Abstract. This report describes the system model that is adopted for state-driven workload generation for a Flight Data Processing System.

1 System model

The system model adopted for state-driven workload generation is showed in Figure 1. Transitions in the Petri Net represent events that can occur during an execution. These events are logged by the Workload Generator during an execution, in order to infer the evolution of the system and to identify whether a target state has been reached. Events are logged by instrumenting the application code in key points, such the entry and exit points of methods of CORBA objects. Table 1 briefly describes the meaning of the considered events: they are related to requests issued by processes in the system, the completion of requests, and lock/unlock operations performed by the Façade on FDP Tables.

In Figure 1, transitions and places are grouped on the basis of their relationships with components of the FDPS (Client, Façade, Processing Server, Load-Balancing Service). The places in the uppermost part of the system model represent the state of the FDP Table in the Façade process: in particular, the number of tokens in places $A1 \dots A6$ represent the number of enqueued requests for the FDP number $1 \dots 6$. When a request is sent from the Façade to a Processing Server, a token is removed from one of the places $B1 \dots B6$, and a token is added in the place $BF1$. In turn, a token is added to one of the places $WRK1, WRK2, WRK3$ according to a load-balancing strategy, which reflect the state of Processing Servers (busy or idle). When a Processing Server finishes, it invokes a callback method of the Façade, which unlocks the FDP and allows the system to process the next pending request for that FDP.

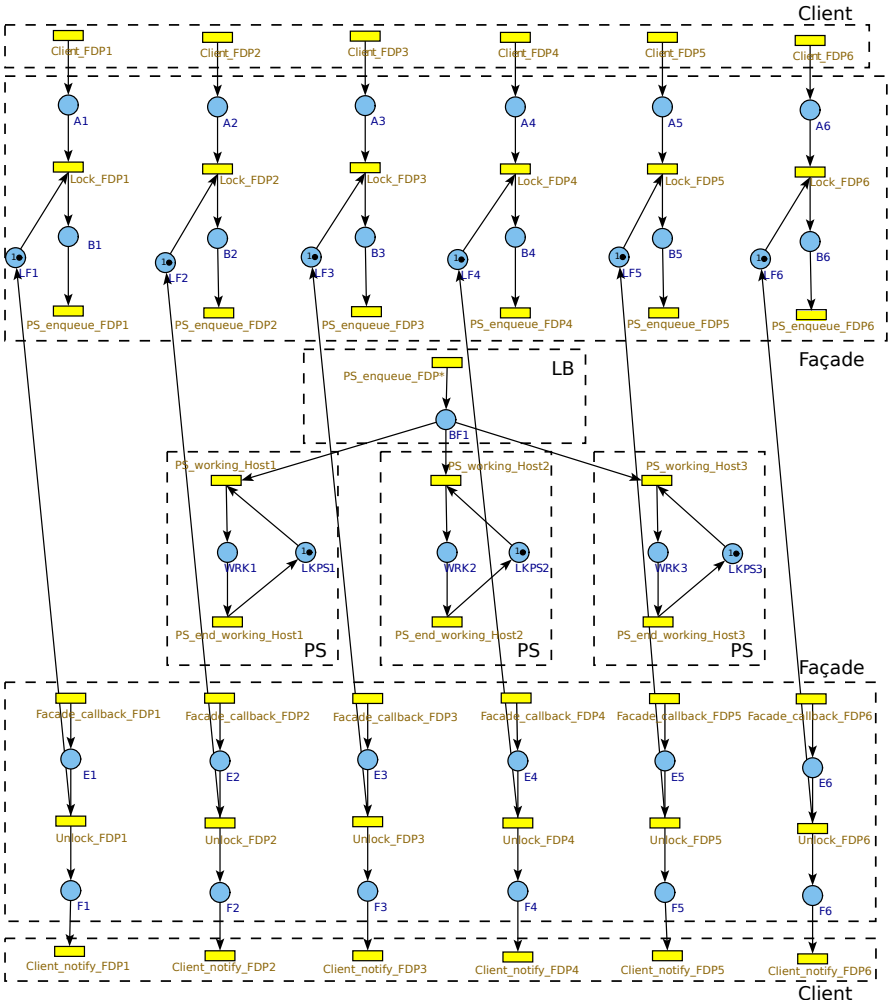

Fig. 1. System model of the FDPS.

Table 1. FDPS events considered in the system model.

Event	**Description**
*Client_FDP**	A client issued a request for an FDP
*Lock_FDP**	The Façade locked an FDP; other requests cannot access the FDP until it is unlocked
*PS_enqueue_FDP**	A request has been forwarded to the group of Processing Servers
*PS_working_Host**	A Processing Server started processing a request
*PS_end_working_Host**	A Processing Server finished processing a request
*Façade_callback_FDP**	The Façade has been notified about the completion of a request
*Unlock_FDP**	The Façade unlocked an FDP
*Client_notify_FDP**	A client has been notified about the completion of a request