18-749 Evaluation Report
Team 2: The House
Chief Experimenter: Joohoon Lee
Members: Paul Cheong, Jun Han, Mohammad Ahmad, Suk Chan Kang

Experimental Setup

- Each client run on separate machine
- Two methods chosen for the experiment (two-way invocation) are highlighted in the Table 1.
  - getPlayerName(String id), getTableName(Integer uid)
  - The reply size is always constant because we always use the same id and uid.
- The middle tier have 2 replicas[1]
- The following parameters are used (Total of 48 combinations)
  - Number of clients: 1,4,7,10
  - Size of reply message: original, 256, 512, 1024 bytes
  - Inter-request time: 0 (no pause), 20, 40ms
- In order to support variable size of reply message, before the client starts invocating 10000 times for the experiment, the client lets the server know the size of the reply for the given run. The server then allocates a dummy data block of the requested size on the entity bean (thus it becomes stateful), and gives that reply when the client invokes the actual method for testing. We used String object to allocate the request data, assuming each character of the String would use 1-byte. Although, this may not be true, since we are interested in size of "useful data", this is acceptable assumption for this evaluation.
- Each test methods are called 5000 times each to make 10000 requests per experiment
- JAVA does not support a fine-grain measurement method, so we decided to use native method using JNI. The probe calls the native implementation of the timer which gives u-sec granularity.
- Total of 7 probes were used in this experiments. The descriptions of each probe are in Table 2.

[1] Our implementation has two replicas and fail-over, but the assumption of this experiment was that there is no fault-tolerance, thus all of our measurements were taken without any crashes on the server.

Table 1: List of Client Invocations[2]

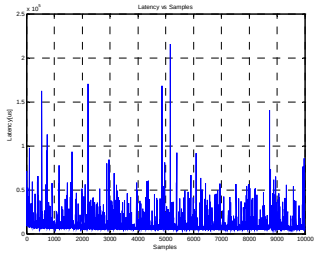
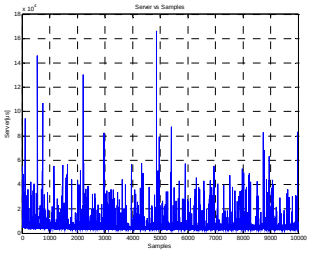
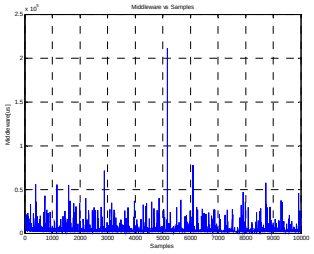
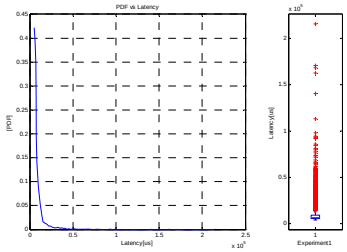
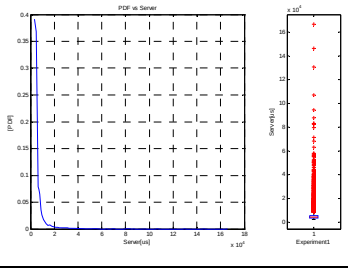
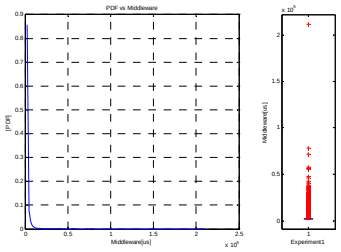
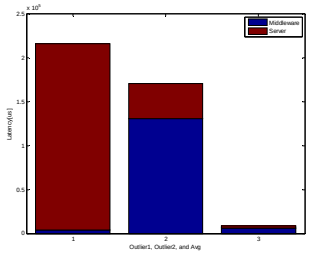
METHOD	ONE_WAY	DB_ACS	SZ_REQ	SZ_REPLY
void savePlayer(int uid, int oid, String id, String name, Integer balance)	YES	YES	12 + sizeof(id)	0
void createTable(int uid, int oid, String name)	YES	YES	8 + sizeof(name)	0
void dealPlayer(int uid, int oid, String playerID)	YES	YES	8 + sizeof(playerID)	0
Card lastDealtCard(String playerID)	NO (2-WAY)	YES	sizeof(playerID)	12 + sizeof(type)
void dealDealer(int uid, int oid, String tableName)	YES	YES	8 + sizeof(tableName)	0
Card lastDealtDealerCard(String tableName)	NO (2-WAY)	YES	sizeof(tableName)	12 + sizeof(type)
void joinTable(int uid, int oid, String playerID, String tableName)	YES	YES	8 + sizeof(playerID) + sizeof(tableName)	0
void placeBet(int uid, int oid, String playerID, int bet)	YES	YES	12 + sizeof(playerID)	0
void startGame(int uid, int oid, String playerID, String tableName)	YES	YES	8 + sizeof(playerID) + sizeof(tableName)	0
void adjustBalance(int uid, int oid, String playerID, int adjust)	YES	YES	12 + sizeof(playerID)	0
int readBalance(String playerID)	NO (2-WAY)	YES	sizeof(playerID)	4
String getPlayerName(String id)	NO (2-WAY)	YES	sizeof(id)	Variable
String getTableName(Integer uid)	NO (2-WAY)	YES	4	Variable

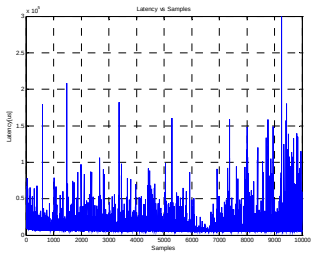
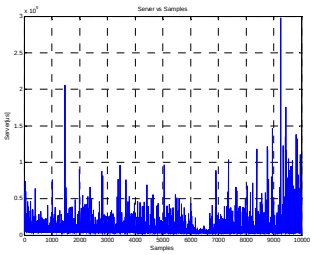
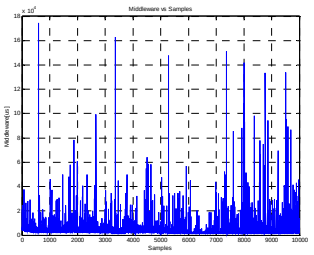
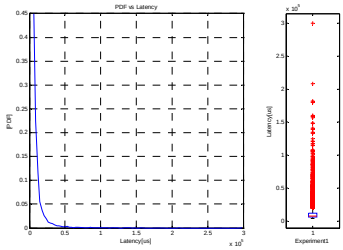
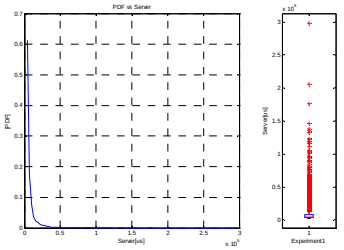
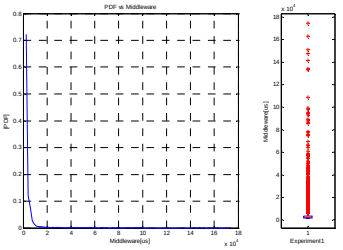
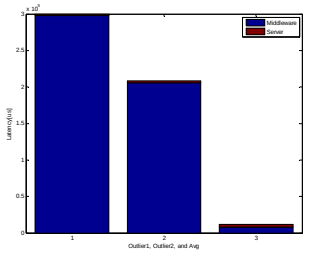
[2] In JAVA, the size of objects cannot be easily determined, so we used simple approximations. Primitive data types are assumed to be 32-bit, and variable size of objects are represented using sizeof(object). The unit of size is 'byte'. When getPlayerName is called for experiment, the id is always the same to fix the reply size

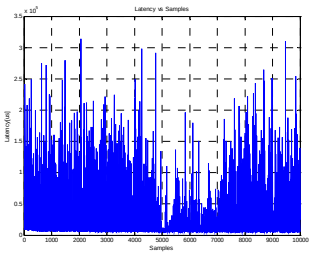
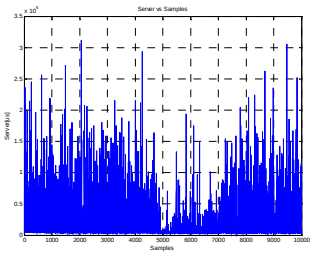
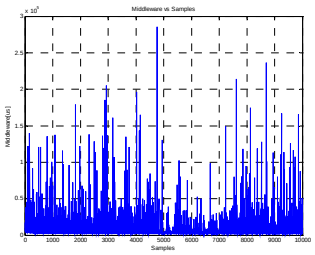
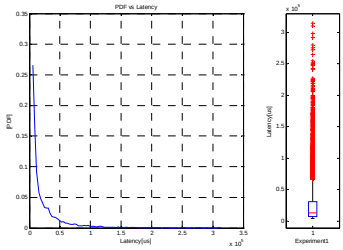
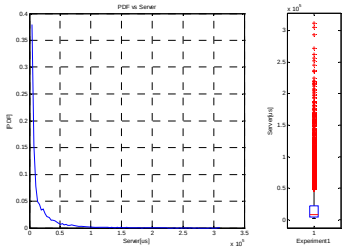
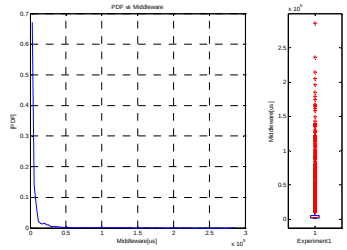
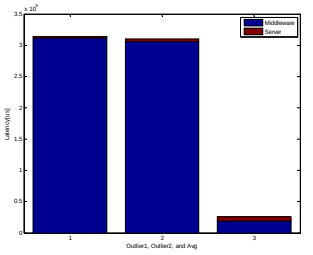
Table 2: List of Probes

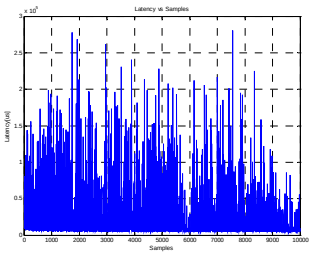
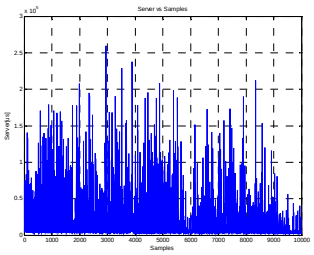
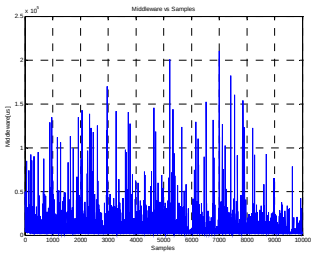
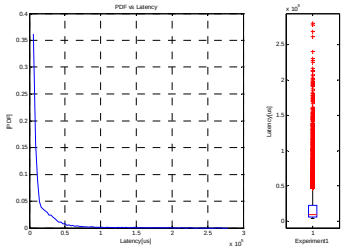
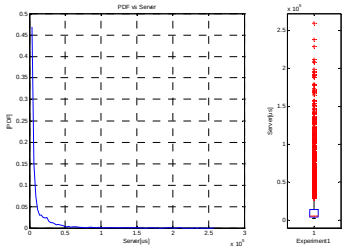
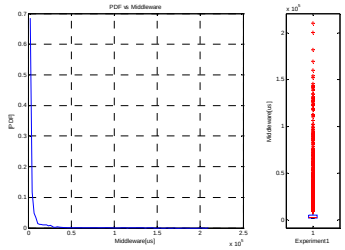
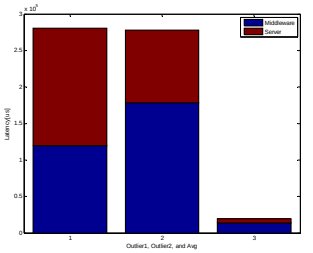
Probe	File Name	Data	Purpose
1	DATA749_app_out_cli_${STY}_2srv_${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Time in u sec	Record the time of client invocation (request going out)
2	DATA749_app_in_cli${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Time in u sec	Record the time of receiving the reply
3	DATA749_app_msg_cli${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Name of each invocation	Keep track of the name of the invocating function to match later
4	DATA749_app_in_srv${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Time in u sec when the request is received	Record the time of request received
5	DATA749_app_out_srv${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Time in u sec when each reply is served	Record the time of request completion
6	DATA749_app_msg_srv${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Name of each invocation	Keep track of the invocation from the server
7	DATA749_app_source_srv${STY}_2srv${C}cli_${IRT}us_${BYT}req_${HOST}_team${N}.txt	Name of client invocation	Keep track of which client sent the each request

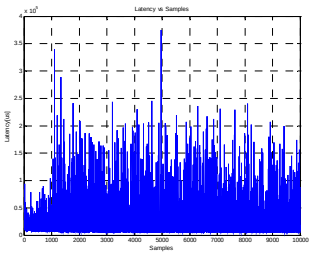
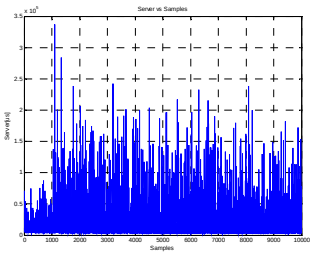
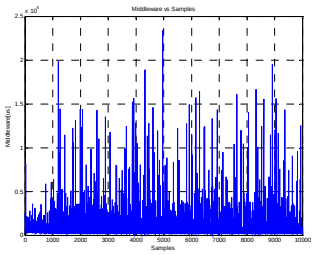
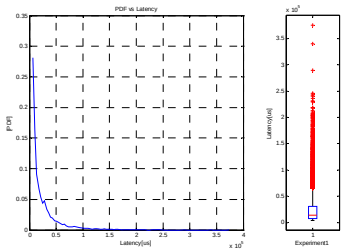
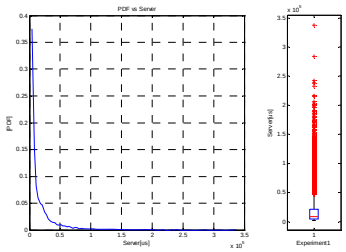
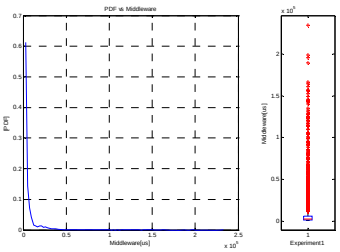
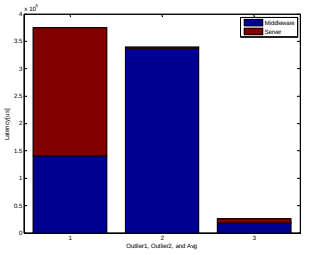
Result

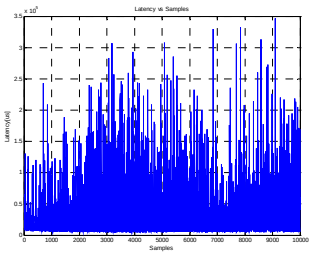
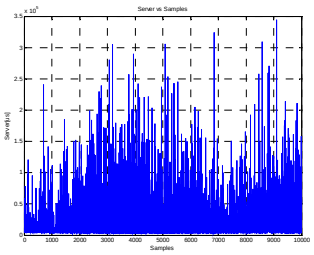
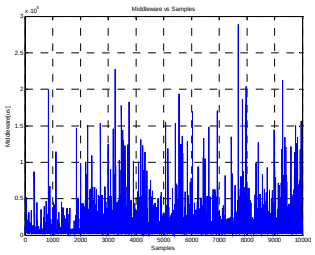
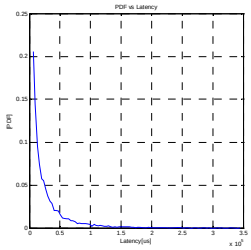
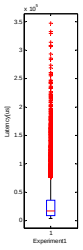
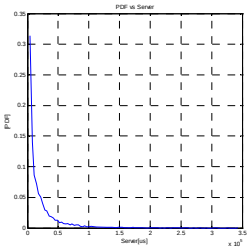
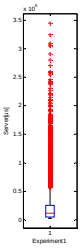
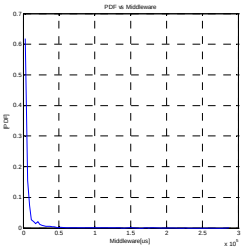
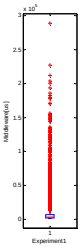
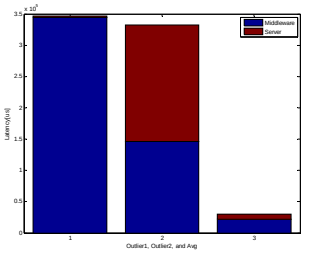
<table>
<tr><td colspan="3">Experiment 1 – # of Client = 1, Size of Reply = original , Request Time = 0ms</td></tr>
<tr><td>Client Latency</td><td>Server Latency</td><td>Middleware Latency</td></tr>
<tr><td> 
</td><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td> 
</td><td> 
</td></tr>
<tr><td> 
</td><td></td><td></td></tr>
</table>

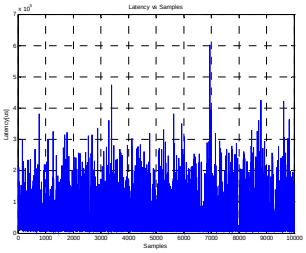
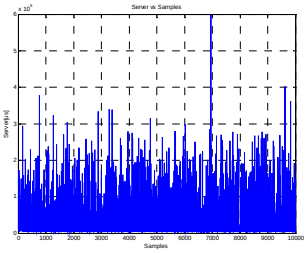
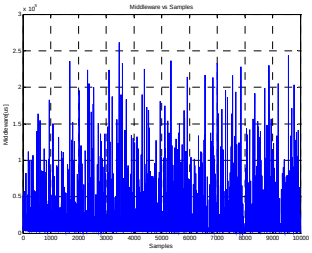
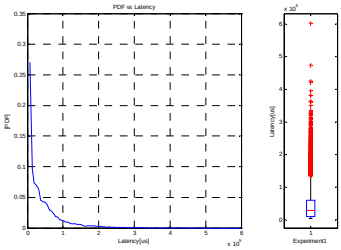
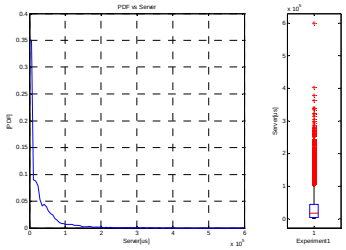
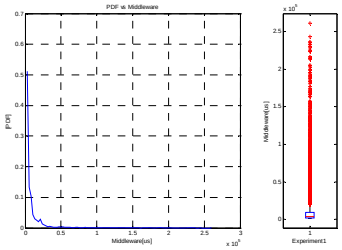
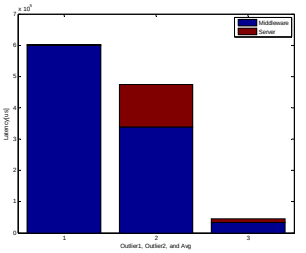
Experiment 2 – # of Client = 1, Size of Reply = original , Request Time = 20ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

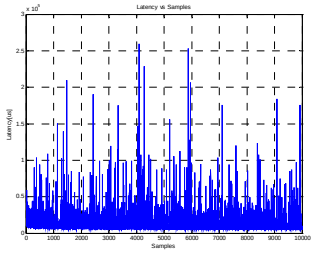
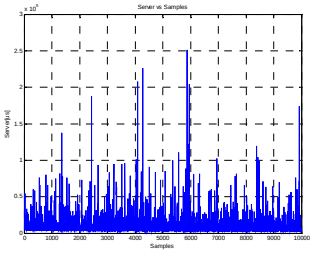
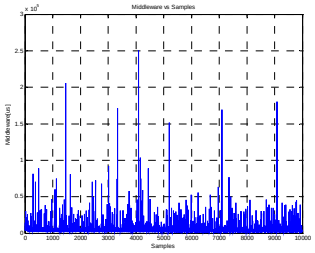
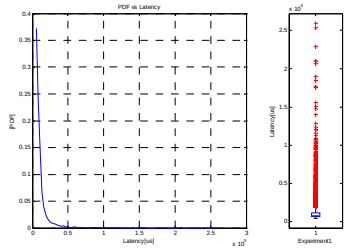
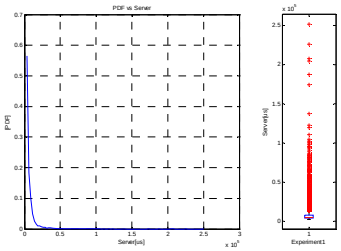
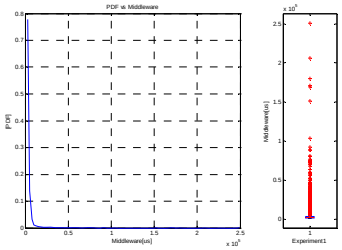
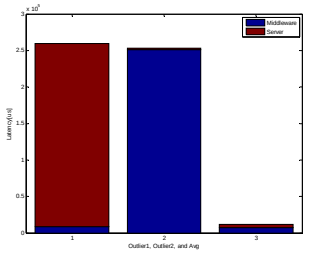
Experiment 3 – # of Client = 1, Size of Reply = original , Request Time = 40ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

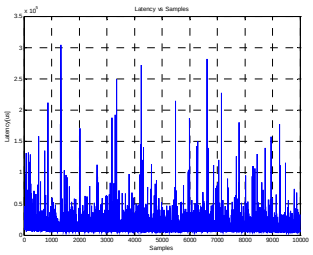
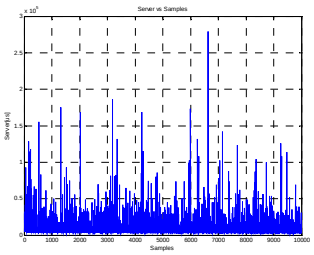
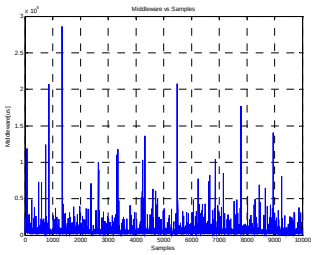
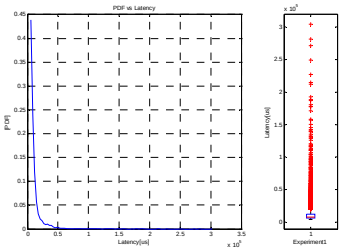
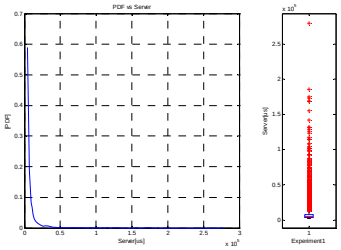
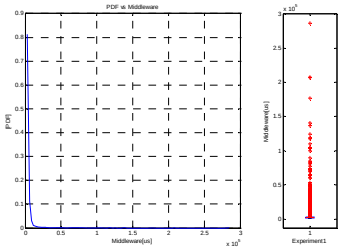
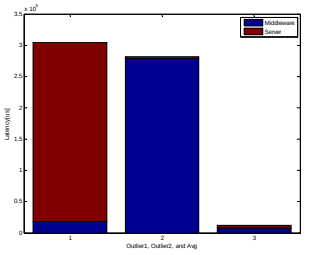
Experiment 4 – # of Client = 1, Size of Reply = 256byte , Request Time = 0ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

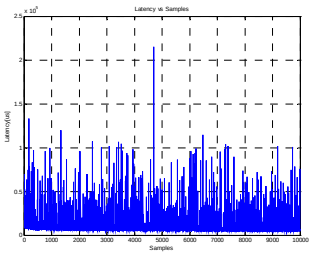
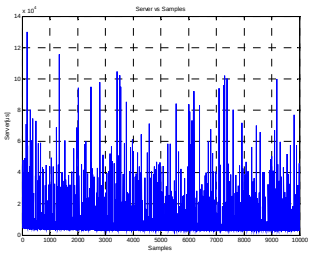
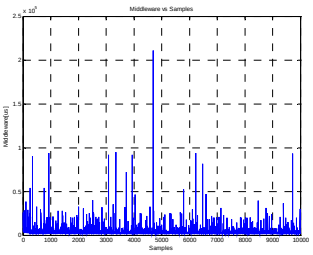
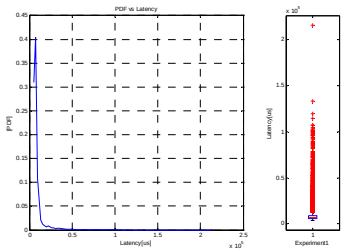
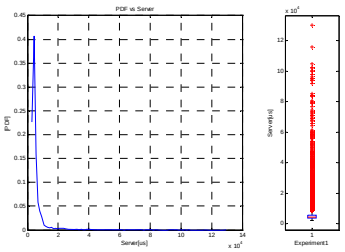
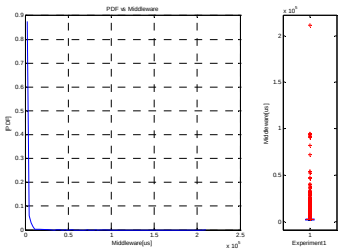
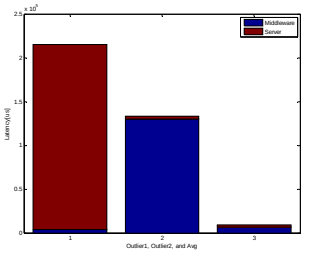
Experiment 5 – # of Client = 1, Size of Reply = 256byte , Request Time = 20ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

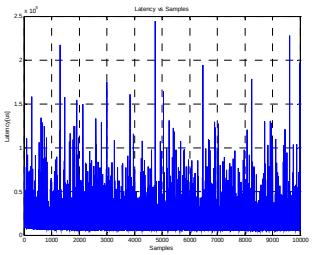
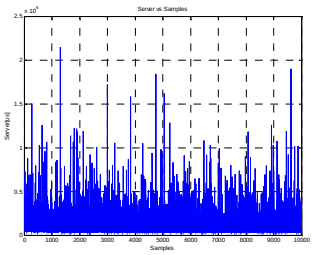
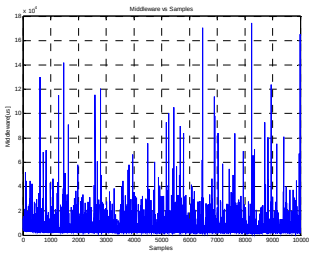
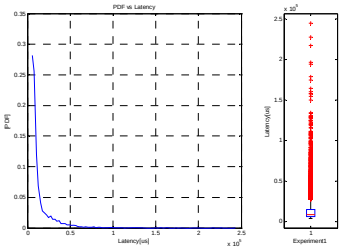
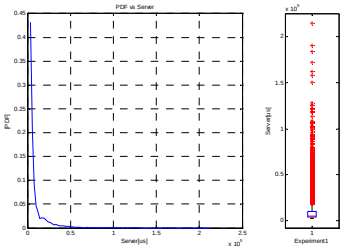
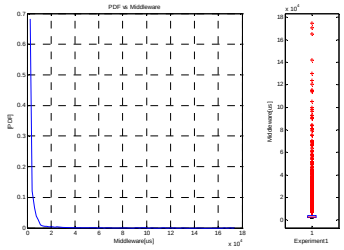
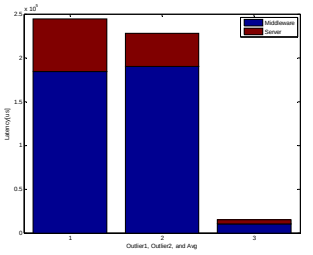
Experiment 6 – # of Client = 1, Size of Reply = 256byte , Request Time = 40ms		
Client Latency	Server Latency	Middleware Latency
		
 	 	 
		

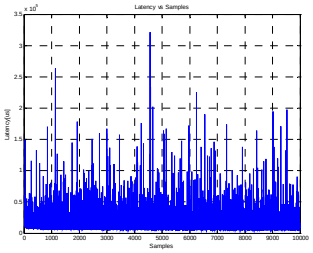
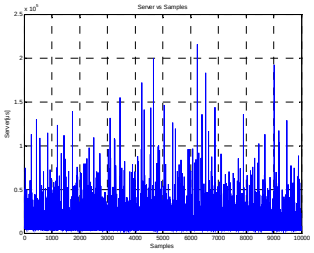
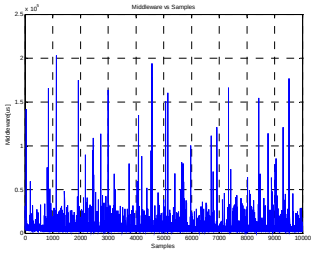
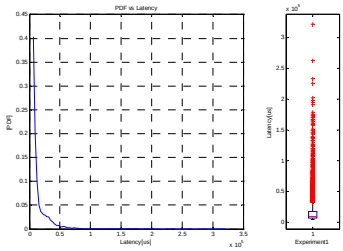
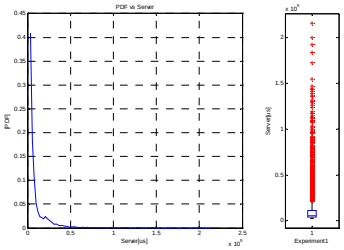
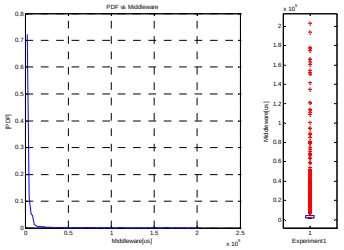
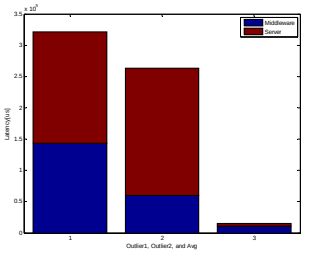
Experiment 7 – # of Client = 1, Size of Reply = 512byte , Request Time = 0ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 8 – # of Client = 1, Size of Reply = 512byte , Request Time = 20ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 9 – # of Client = 1, Size of Reply = 512byte , Request Time = 40ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 10 – # of Client = 1, Size of Reply = 1024byte , Request Time = 0ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 11 – # of Client = 1, Size of Reply = 1024byte , Request Time = 20ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 12 – # of Client = 1, Size of Reply = 1024byte , Request Time = 40ms		
Client Latency	Server Latency	Middleware Latency
		
		
		

Experiment 1~12 – # of Client = 1 (Overall Statistics)		
Client Latency	Server Latency	Middleware Latency