A New Testing Platform and Client-Server Game Theory Applied in the RHIC Control System

Yuan Gao

Department of Electrical and Computer Engineering

Outlines

PART 1:
A simulation platform for testing system code reliability

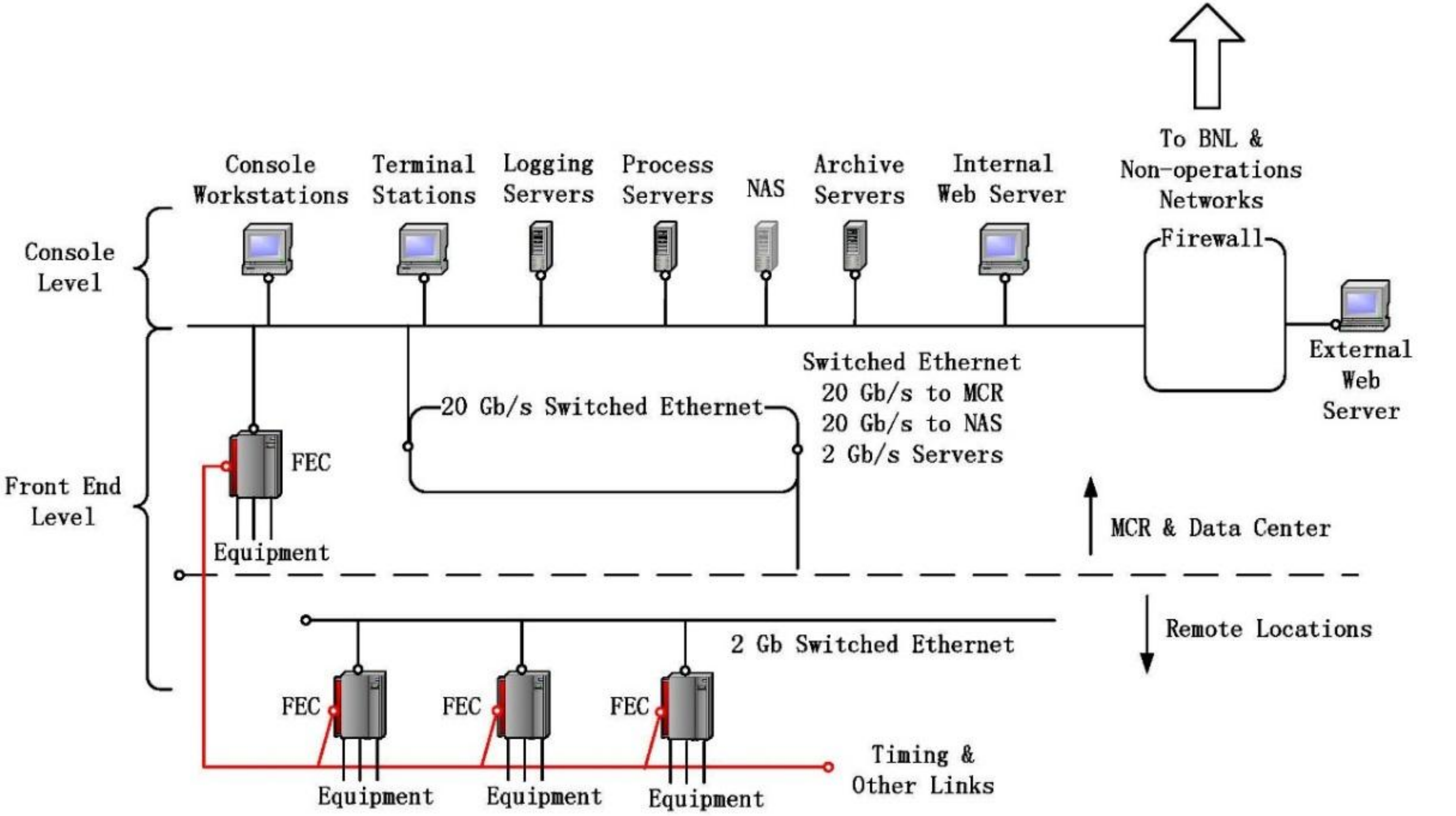
PART 2:
Applying Game Theory for solving a practical system problem

Outlines

PART 1:
A simulation platform for testing system code reliability

PART 2:
Applying Game Theory for solving a practical system problem

Accelerator Control System Overview



System Components and Tools

Components:

- Accelerator Device Object (ADO)
- Controls Name Server (CNS)
- Logging system…
- Notification server…

Tools:

- Parameter Editing Tool (PET)
- Logging Data Display Tools: Gpm, LogView…

System Components and Tools

Components:

- ➢ Accelerator Device Object (ADO)
- ➢ Controls Name Server (CNS)
- ➢ Logging system…
- ➢ Notification server…

- A fundamental conception;
- Controls software system is built on it;
- ADO data can be viewed or edited by PET.

Tools:

- ➢ Parameter Editing Tool (PET)
- ➢ Logging Data Display Tools: Gpm, LogView…

System Components and Tools

Components:

- ➢ Accelerator Device Object (ADO)
- ➢ Controls Name Server (CNS)
- ➢ Logging system…
- ➢ Notification server…

- Work similarly to a DNS;
- Store unique name/value pairs, so that requested data can be accessed.

Tools:

- ➢ Parameter Editing Tool (PET)
- ➢ Logging Data Display Tools: Gpm, LogView…

System Components and Tools

Components:

- ➢ Accelerator Device Object (ADO)
- ➢ Controls Name Server (CNS)
- ➢ Logging system…
- ➢ Notification server…

- Log accelerator data from previous runs;
- Post-mortem analysis;
- Tools available for creating/editing logging requests, starting/stopping logging process, viewing logged data.

Tools:

- ➢ Parameter Editing Tool (PET)
- ➢ Logging Data Display Tools: Gpm, LogView…

System Components and Tools

Components:

- Accelerator Device Object (ADO)
- Controls Name Server (CNS)
- Logging system…
- Notification server…

Receive notifications, log notices in a daily log and forward them to generate alarm.

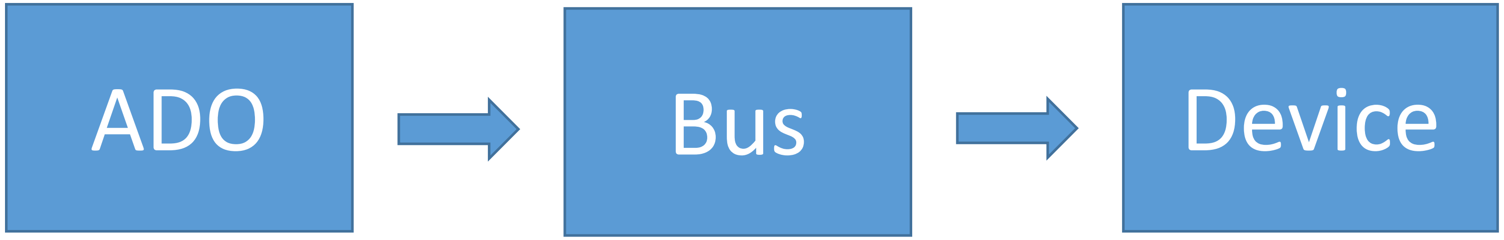
Tools:

- Parameter Editing Tool (PET)
- Logging Data Display Tools: Gpm, LogView…

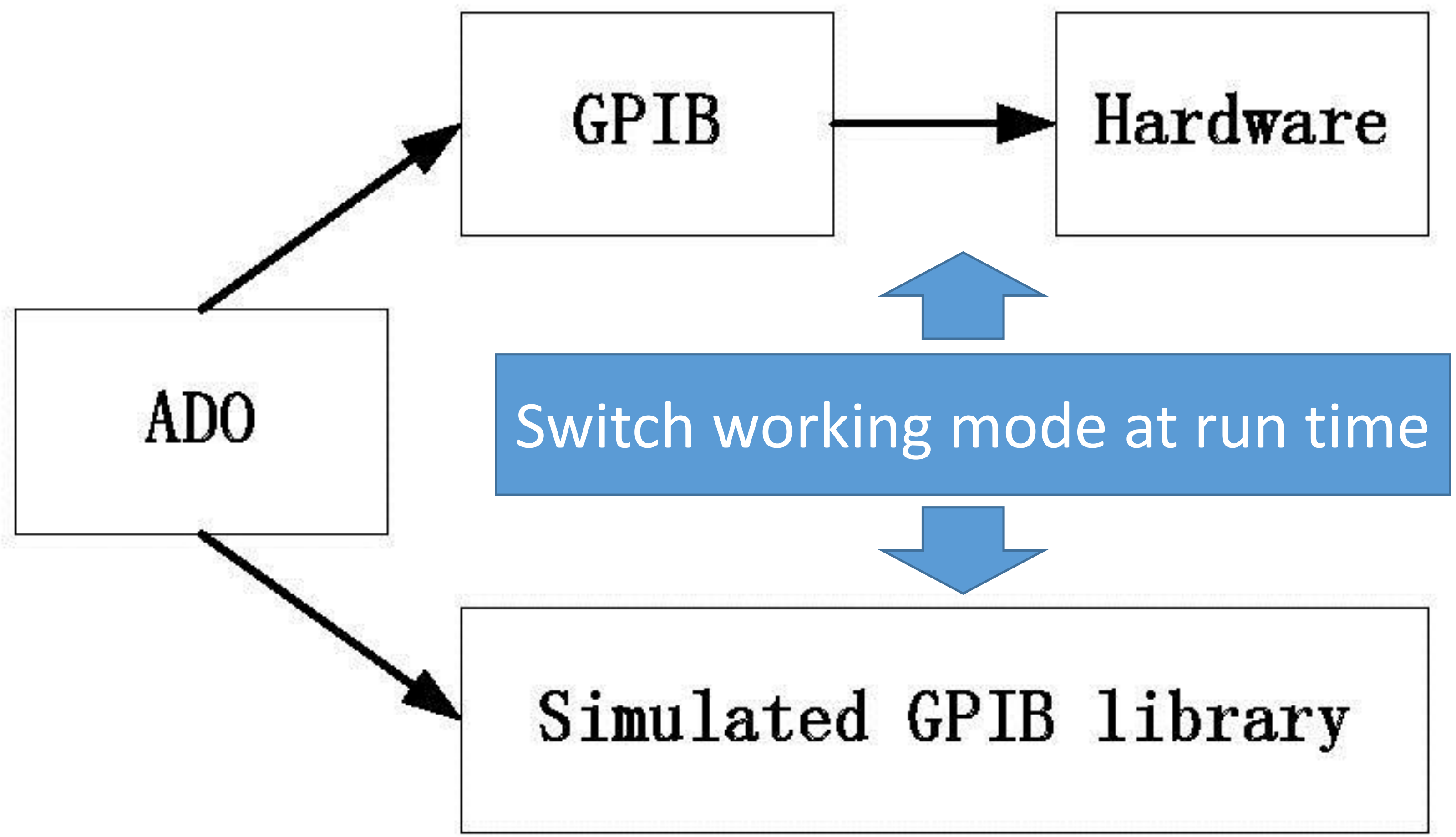
Why Testing Platform?

To improve ADO codes reliability

Real-world communication of an ADO:

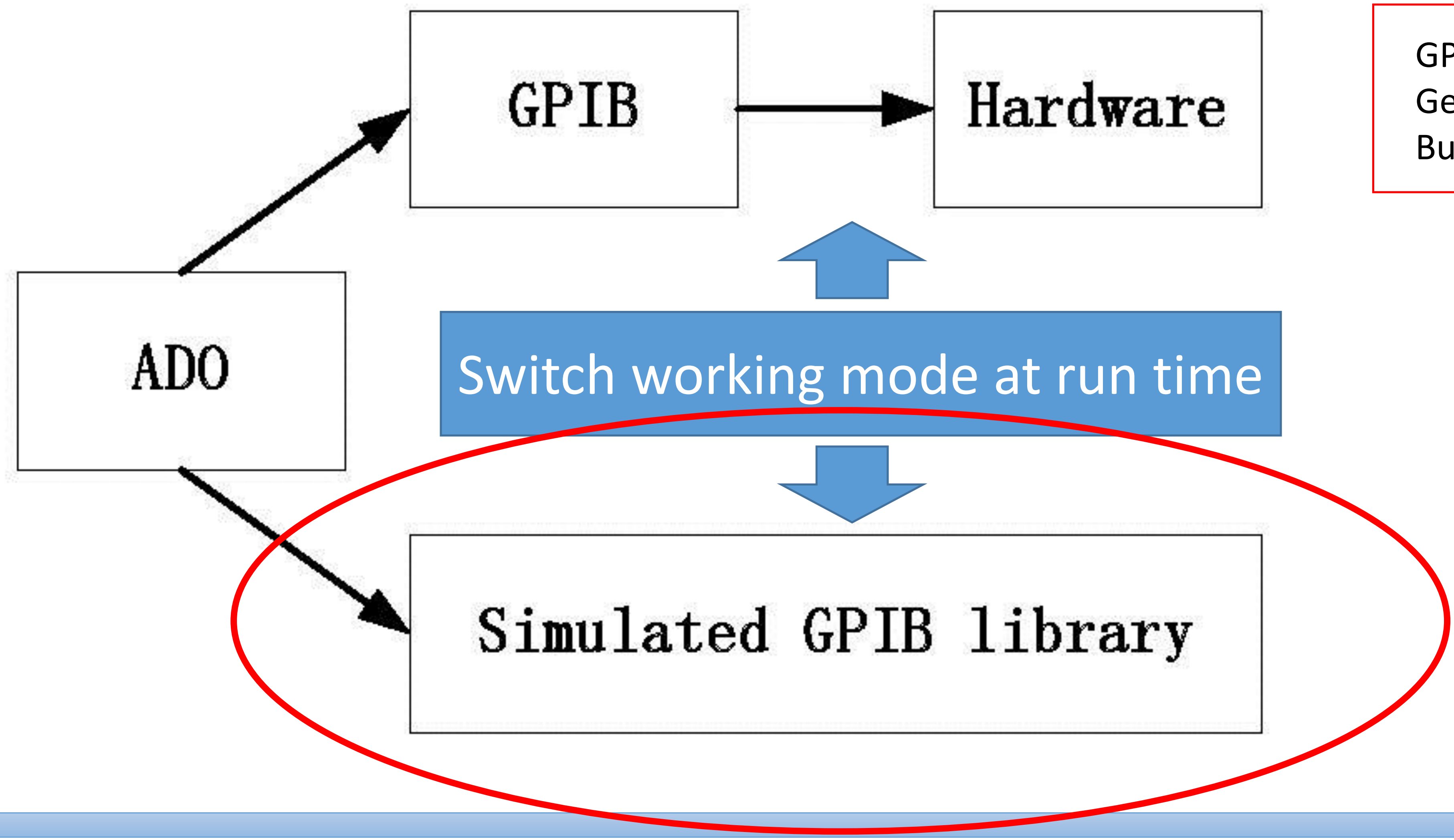


Basic Structure



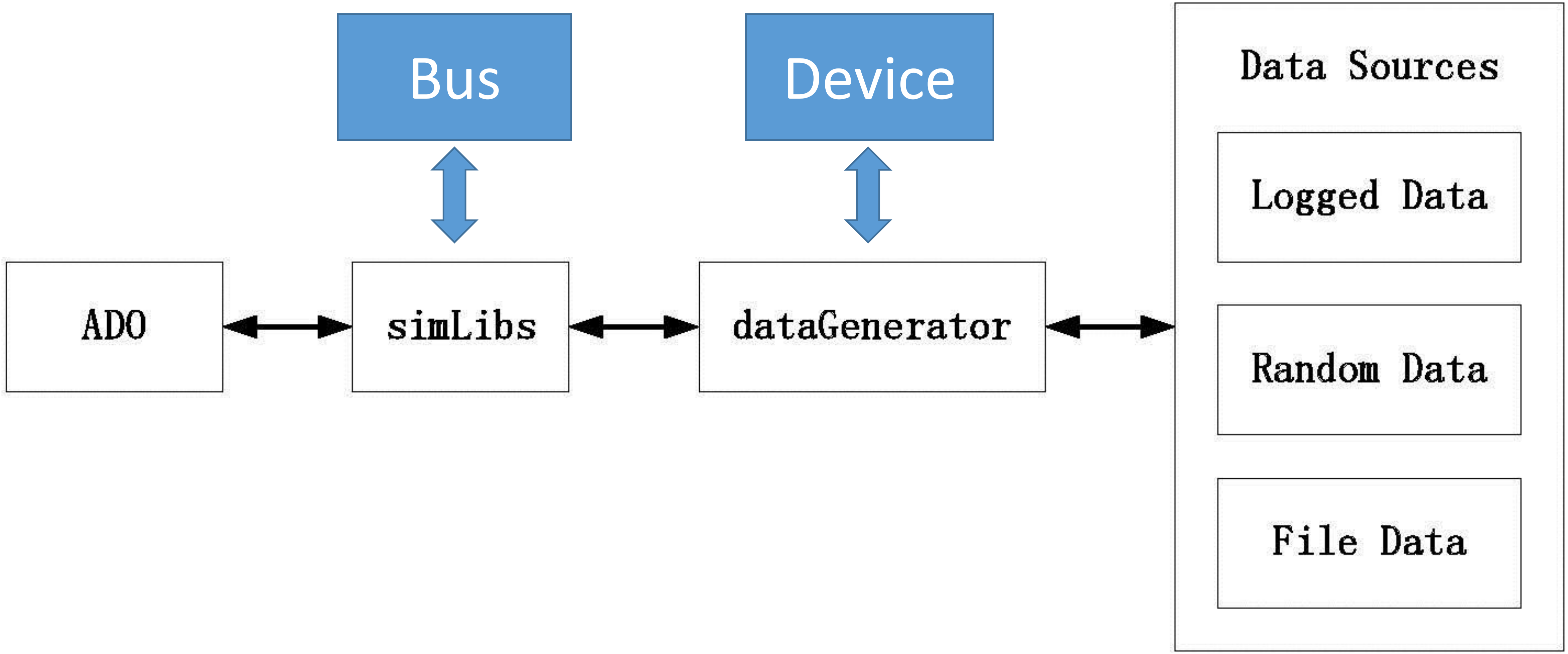
GPIB:
General Purpose Interface Bus, , IEEE 488.2 standard

Basic Structure



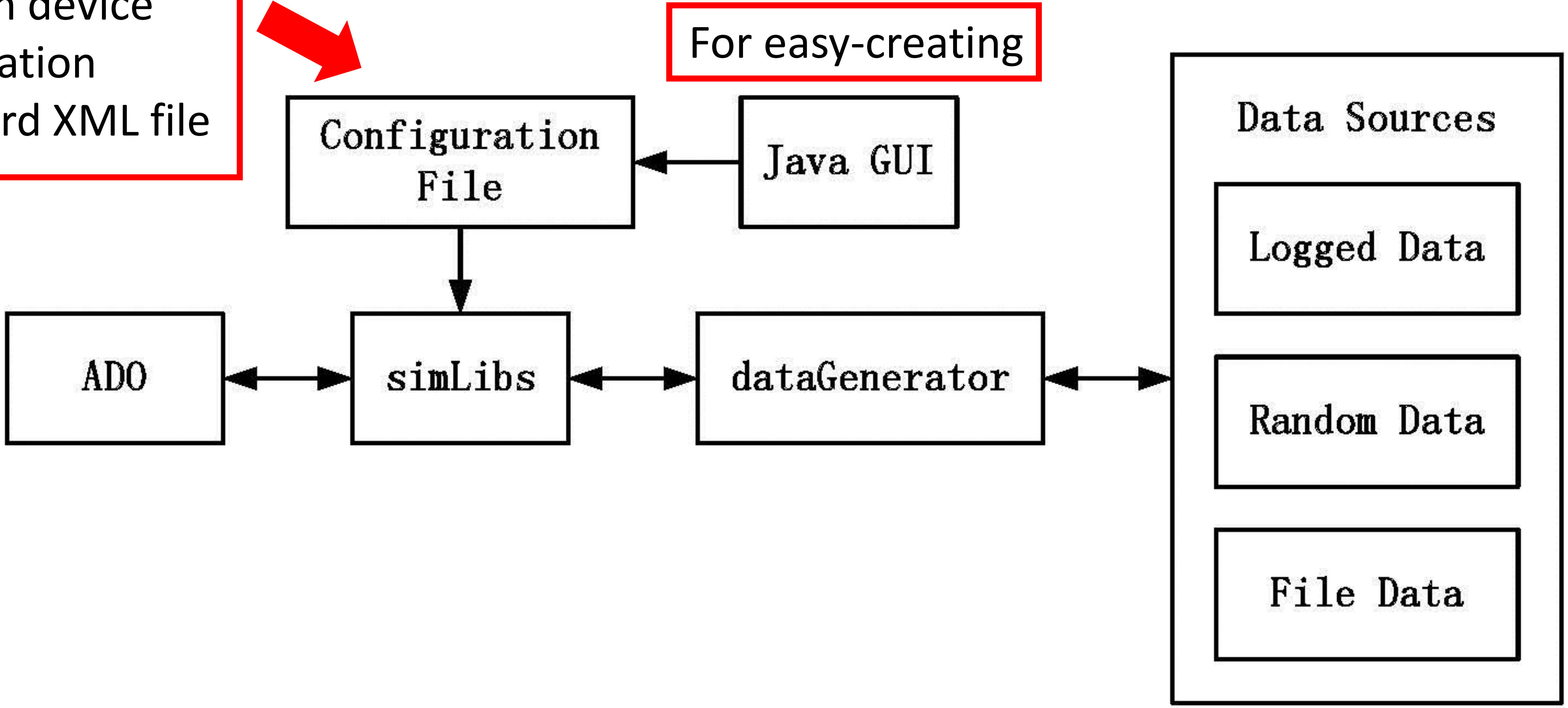
GPIB:
General Purpose Interface Bus, , IEEE 488.2 standard

Basic Structure in Details – Two main blocks

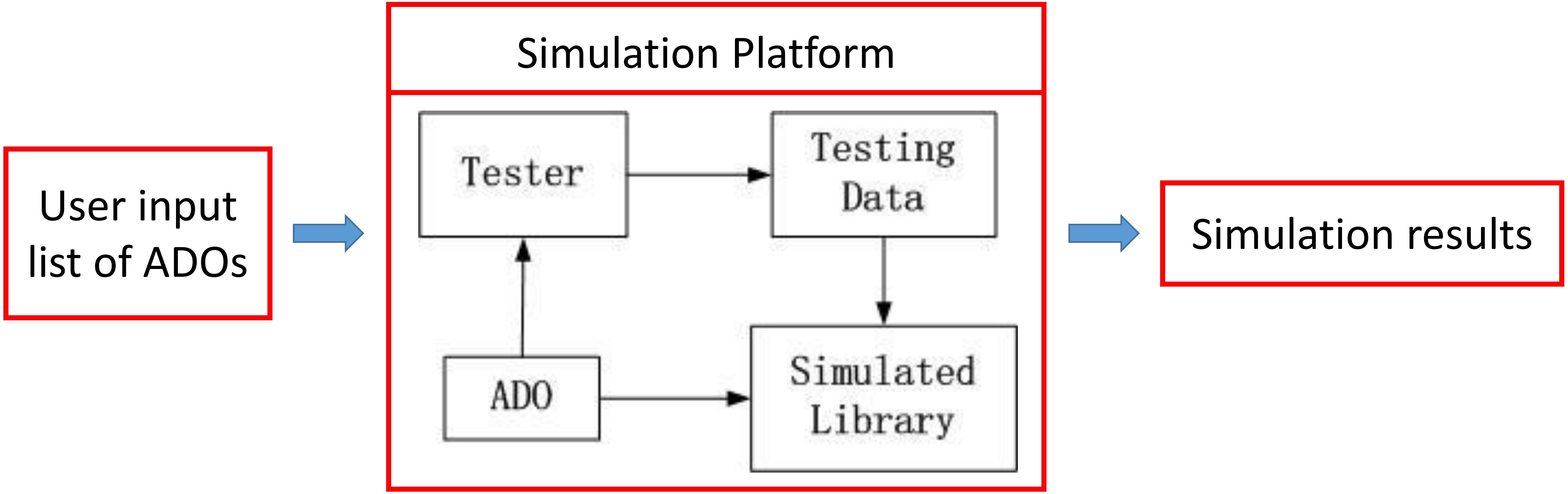


Generalization – To accommodate various types of devices

- Contain device information
- Standard XML file

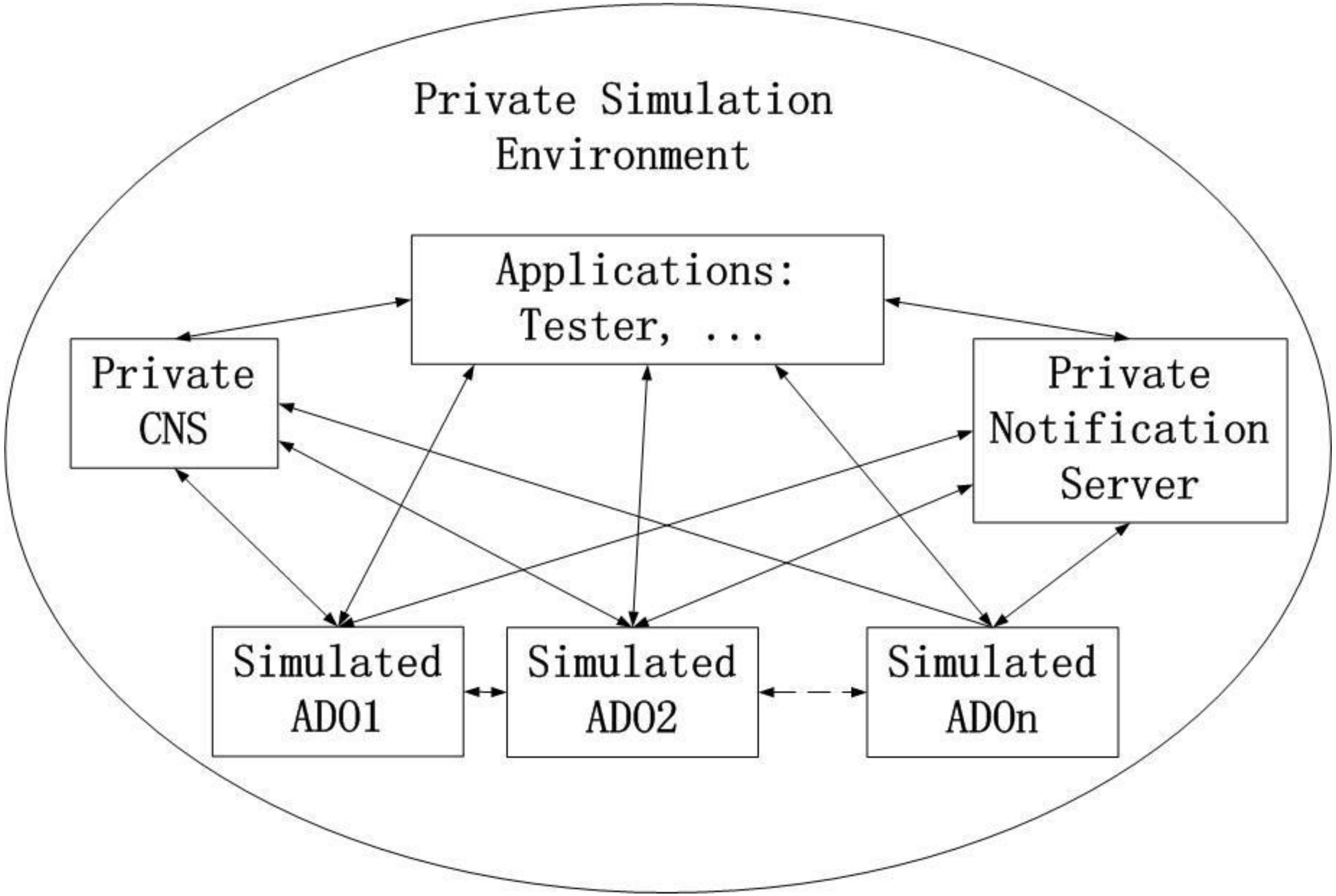


Testing Automation – Free the users



What is next?

Build a private simulated environment for each developer, containing private CNS/notification server, and simulated ADOs. Each simulated environment is independent between each other and outside system, and is user-customizable.



Key Use Cases

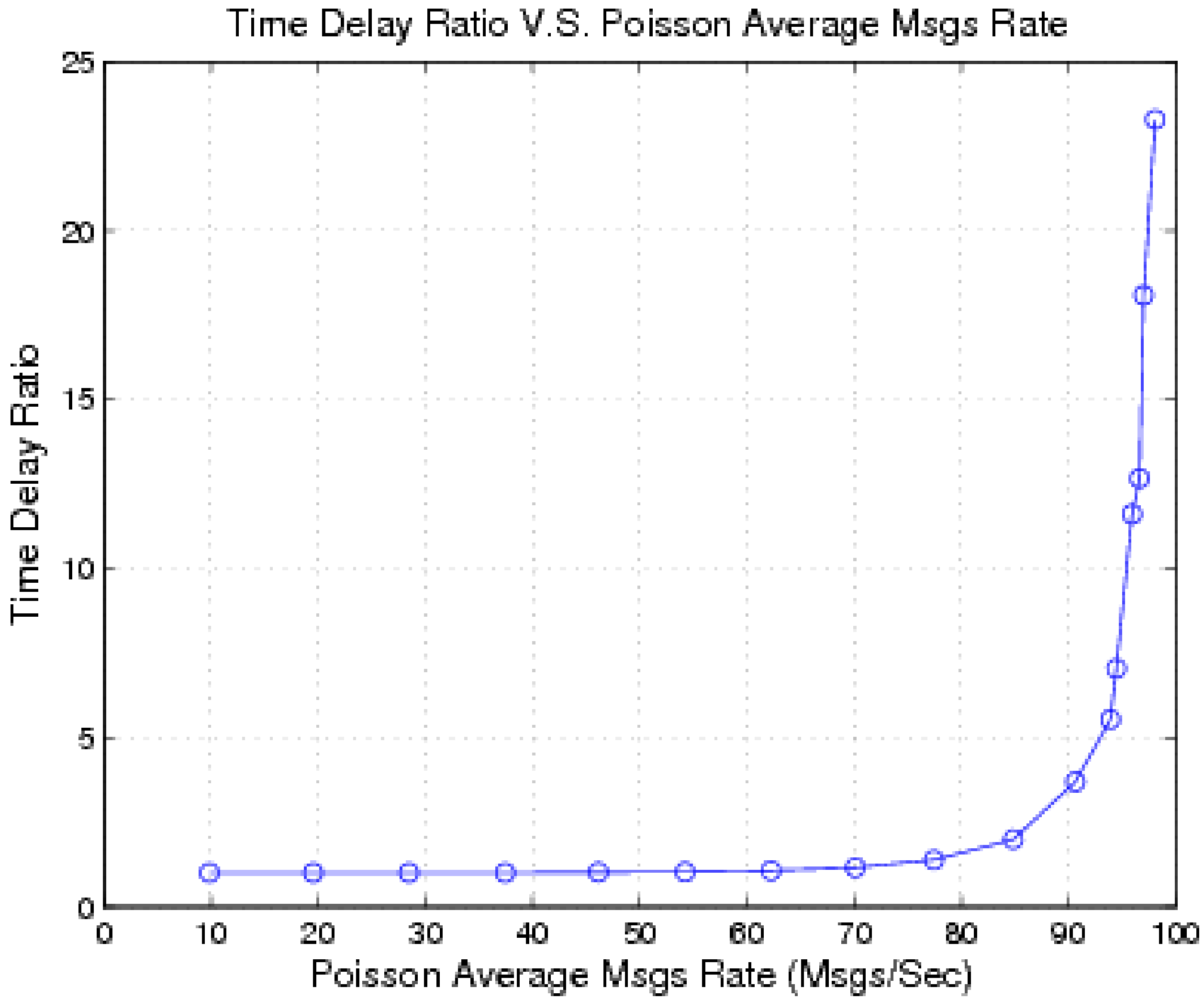
- Improve robustness of ADO codes by running testing data.
- Verify upgrade of software, whether the new version of software works in a desired way.
- Replace real hardware when they are not available.
- Specialized testing, control parameterization method.

PART 1:
A simulation platform for testing system code reliability

PART 2:
Applying Game Theory for solving a practical system problem

What the Problem?

Client-Server Problem:
In the RHIC front end system, every computer acts as a server providing services to a large number of clients. When the number of clients reaches its limit, the system slows down or even crashes.



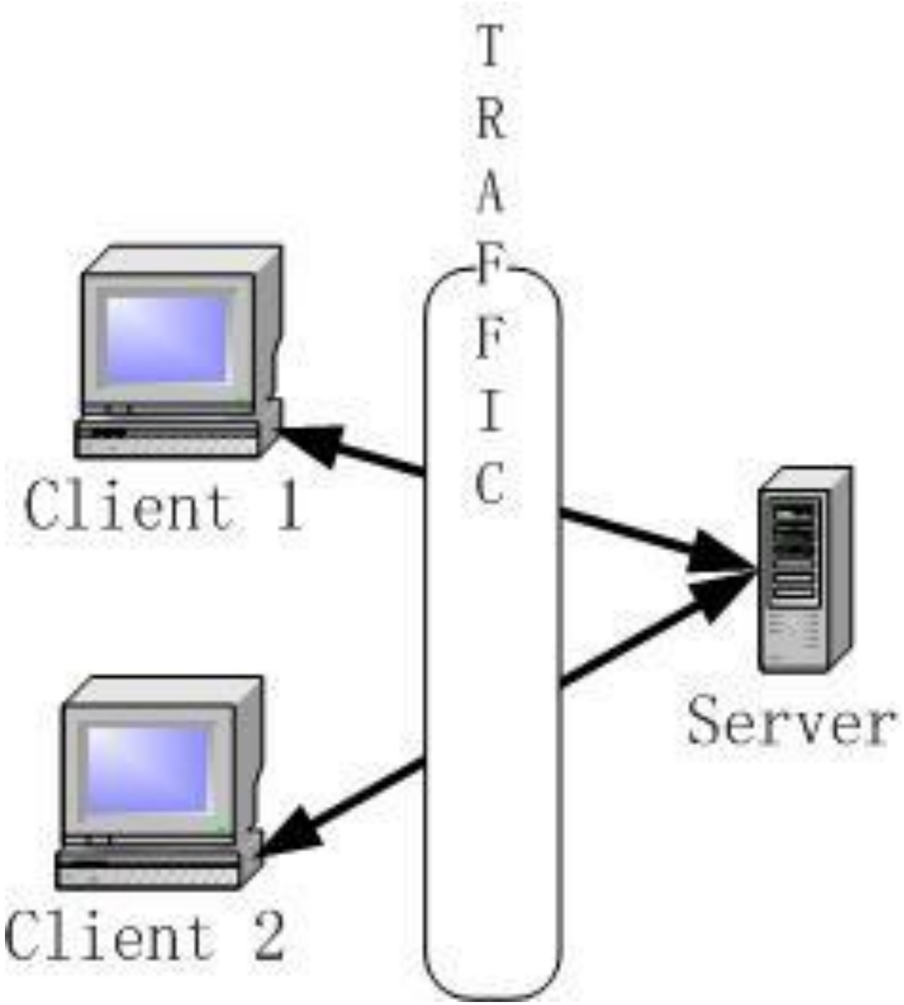
Why Game Theory?

"Game theory aims to help people understand situations in which decision-makers interact."
- Martin J. Osborne

What decision-makers? → Clients and server

What interactions? → Traffic interactions

A simple example of a client-server game



Game Rules

- 2 clients are talking to 1 server, each has 10 traffic;
- Server can handle any amount of traffic from any one of them, but not both;
- If a client's transmission is successful, that client gets a profit = it's amount of traffic = 10;
- However, if both of them send requests at the same time, server crashes, both of them get a punishment = -c = -10;
- Clients can always choose being idle, in which case, profit = 0 will be assigned.

A simple example of a client-server game

	Client 2 Sends	Client 2 Holds
Client 1 Sends	Client 1 gets -10 Client 2 gets -10	Client 1 gets 10 Client 2 gets 0
Client 1 Holds	Client 1 gets 0 Client 2 gets 10	Client 1 gets 0 Client 2 gets 0

Pure Strategy Nash Equilibrium

Nash Equilibrium (NE) is a strategy profile, such that, no players can better off by singly changing action, given that all the other players stick to their actions.

A simple example of a client-server game

	Client 2 Sends	Client 2 Holds
Client 1 Sends	**Client 1 gets -10** **Client 2 gets -10**	**Client 1 gets 10** **Client 2 gets 0**
Client 1 Holds	Client 1 gets 0 Client 2 gets 10	Client 1 gets 0 Client 2 gets 0

Pure Strategy Nash Equilibrium

Pure strategy NE: Each player chooses only 1 action;
Mixed strategy NE: Players can randomize among their available actions.

A pure strategy NE = A mixed strategy NE with each player assigns probability 1 to one of their available actions

A simple example of a client-server game

	Client 2 Sends	Client 2 Holds
Client 1 Sends	Client 1 gets -10 Client 2 gets -10	Client 1 gets 10 Client 2 gets 0
Client 1 Holds	Client 1 gets 0 Client 2 gets 10	Client 1 gets 0 Client 2 gets 0

Pure Strategy Nash Equilibrium

	1/2 time Sends	1/2 time Holds
1/2 time Sends	1/4	1/4
1/2 time Holds	1/4	1/4

Mixed Strategy Nash Equilibrium

Expected payoff = 0 for both clients

A simple example of a client-server game

Can we do better?

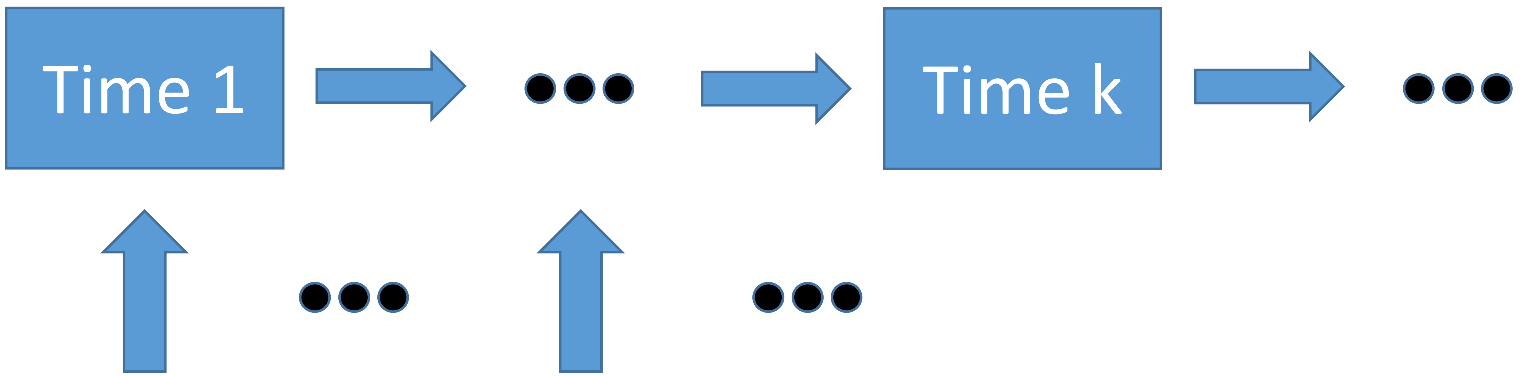
	Client 2 Sends	Client 2 Holds		
Client 1 Sends	**Client 1 gets -10** **Client 2 gets -10**	**Client 1 gets 10** **Client 2 gets 0**	**0**	**1/2**
Client 1 Holds	Client 1 gets 0 Client 2 gets 10	Client 1 gets 0 Client 2 gets 0	1/2	0

Yes!
By using signals:
For example, flip a fair coin:
If Head: client 1 Sends, client 2 Holds;
If Tail: client 1 Holds, client 2 Sends.

Correlated Equilibrium
Expected payoff = (10+0)/2 = 5
for both clients

Client-Server Game Model – Repeated Game

Repeated game:
A same stage game is played over and over again.



Stage Game

Players	A set of n clients
Actions	Send (S) or Hold (H)
Client i's traffic	t_i
Server crash punishment	-c

	S	H
S	-c, -c	t1, 0
H	0, t2	0, 0

Payoff table of a 2-client example

Subgame Perfect Equilibrium – Pure Strategy

Subgame Perfect Equilibrium (SPE) is a Nash equilibrium in every subgame, every finite horizon game admits at least one SPE, can be calculated by backward induction.

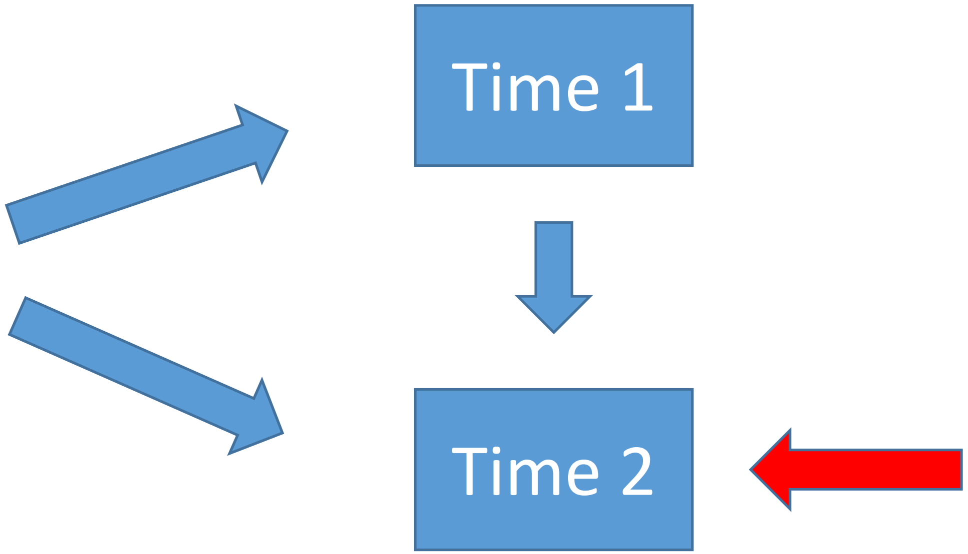
- Subgame: A game follows any history;
- Finite horizon game: Every player has a finite number of actions;
- Backward induction: Calculate Nash equilibrium from last stage, and roll-back to first stage. Thus, it can only apply to a game with finite number of stages.

Subgame Perfect Equilibrium – Pure Strategy

Subgame Perfect Equilibrium (SPE) is a Nash equilibrium in every subgame, every finite horizon game admits at least one SPE, can be calculated by backward induction.

2-client Stage Game

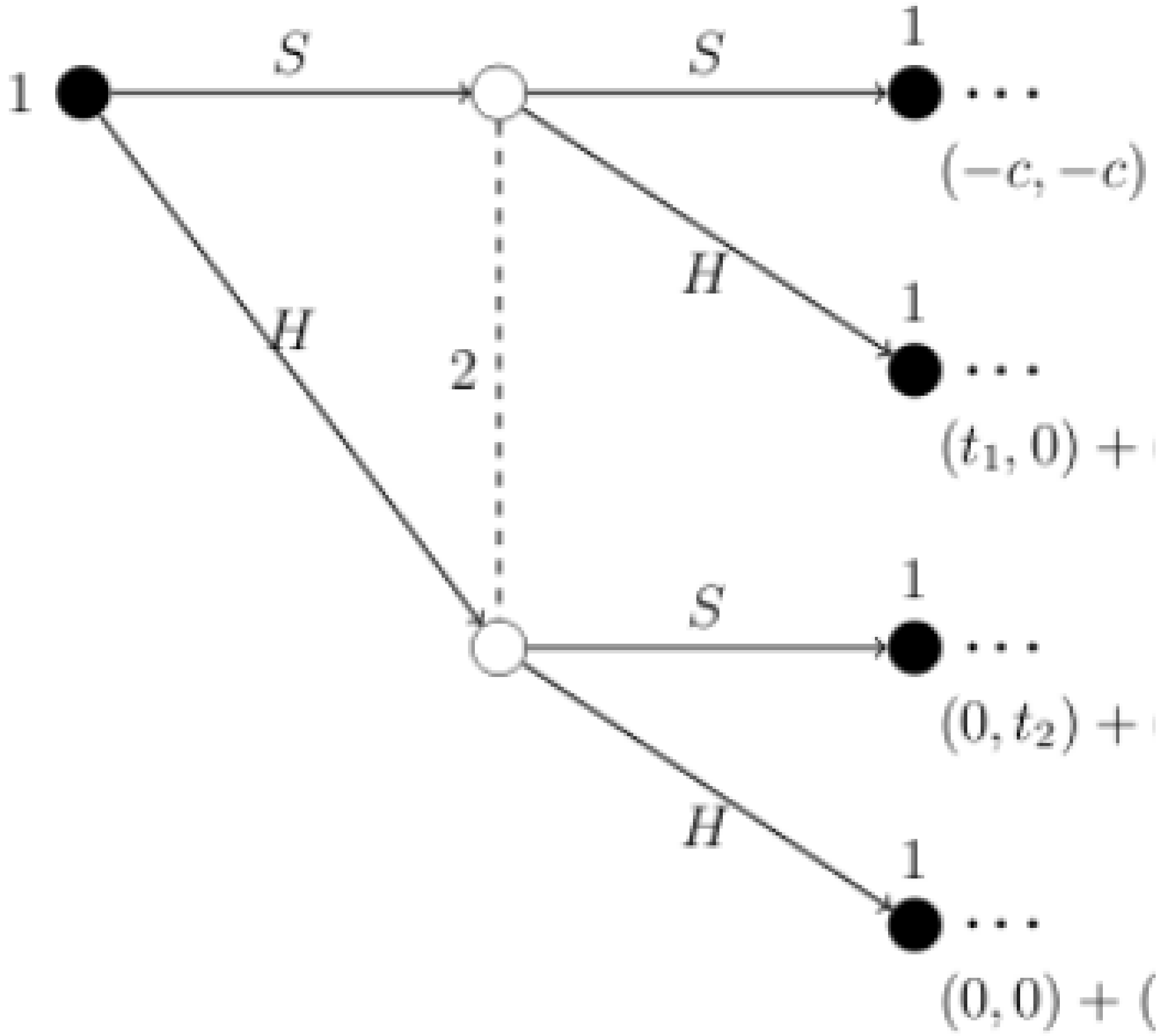
	S	**H**
S	-c, -c	t1, 0
H	0, t2	0, 0



Subgame Perfect Equilibrium – Pure Strategy

Time 2

	S	H
S	?	?
H	?	?



Subgame Perfect Equilibrium – Pure Strategy

Time 2

	S	H
S	?	?
H	?	?

=

Stage 1 game payoffs

(Pay1, Pay2)

+

Current stage game payoffs

	S	H
S	-c, -c	t1, 0
H	0, t2	0, 0

Subgame Perfect Equilibrium – Pure Strategy

Time 2

	S	H
S	?	?
H	?	?

=

Stage 1 game payoffs

(Pay1, Pay2)

+

Current stage game payoffs

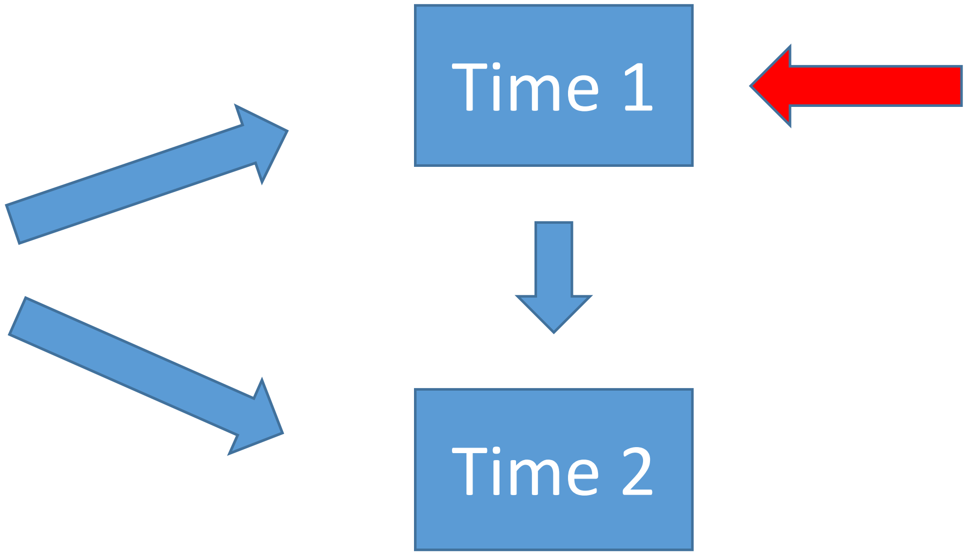
	S	H
S	-c, -c	t1, 0
H	0, t2	0, 0

Subgame Perfect Equilibrium – Pure Strategy

Subgame Perfect Equilibrium (SPE) is a Nash equilibrium in every subgame, every finite horizon game admits at least one SPE, can be calculated by backward induction.

2-client Stage Game

	S	**H**
S	-c, -c	t1, 0
H	0, t2	0, 0



Subgame Perfect Equilibrium – Pure Strategy

Time 1

	S	H
S	?	?
H	?	?

=

Current stage game payoffs

	S	H
S	-c, -c	t1, 0
H	0, t2	0, 0

\+

Stage 2 game equilibrium payoffs

(Pay1, Pay2)

Subgame Perfect Equilibrium – Pure Strategy

Time 1

	S	H
S	?	?
H	?	?

=

Current stage game payoffs

	S	H
S	-c, -c	t1, 0
H	0, t2	0, 0

+

Stage 2 game equilibrium payoffs

{ Pay1, Pay2 }

Subgame Perfect Equilibrium – Pure Strategy

The pure strategy SPE for a 2-client 2-period game is that in each stage, exact 1 client chooses "Send".

↓

The pure strategy SPE for a n-client k-period game is that in each stage, exact 1 client chooses "Hold", the rest choose "Send", k can be finite or infinite.

Subgame Perfect Equilibrium – Mixed Strategy

Indifference Principle:

If in an equilibrium players' strategies are mixing, they must be indifferent between their strategies.

↓

Expected payoff of "Send" = Expected payoff of "Hold" = 0

Subgame Perfect Equilibrium – Mixed Strategy

Expected payoff of "Send" = Expected payoff of "Hold" = 0

$$(-c)\prod_{\substack{j\in \boldsymbol{N}\\ j\neq i}} p_j + t_i\Big(1-\prod_{\substack{j\in \boldsymbol{N}\\ j\neq i}} p_j\Big) = 0, \forall i \in \boldsymbol{N}$$

$$p_i = \sqrt[n-1]{\frac{(t_i+c)^{n-2}\prod_{k\in \boldsymbol{N}, k\neq i} t_k}{t_i^{n-2}\prod_{k\in \boldsymbol{N}, k\neq i}(t_k+c)}}, \forall i \in \boldsymbol{N}$$

Check if all p_i are in [0, 1]

If yes, then the mixed strategy SPE is that in every stage, every client plays "Send" with probability "p_i"

What is our purpose?

Use game theory to solve the client-server problem

↓

Design game dynamics leading clients to paly equilibrium

What is our purpose?

Game dynamics design

Nash equilibrium

- There are no general natural dynamics leading to Nash equilibria [Hart, 2011].
  - “general”: in all games;
  - “natural”: adaptive, simple and efficient;
  - “leading to Nash equilibria”: at a Nash equilibrium (or close to it) from some time on.
- Lower bounds that are exponential in the number of players for the communication complexity in each of the following cases [Hart and Mansour, 2010]:
  - Reaching a pure Nash equilibrium;
  - Reaching a pure Nash equilibrium in a Bayesian setup;
  - Reaching a mixed Nash equilibrium.

What is our purpose?

Game dynamics design

Nash equilibrium → Hard!

What is our purpose?

Game dynamics design

Nash equilibrium → Hard!

Correlated equilibrium ?

What is our purpose?

Game dynamics design

Nash equilibrium

Correlated equilibrium

- [Hart and Mansour, 2010] shows that the communication complexity of reaching a correlated equilibrium is polynomial in the number of players.
- "Regret matching procedure" [Hart and Mas-Colell, 2000, 2001]:
  - "Regret": the increase in past payoff, if any, if a different action would have been used;
  - "Matching": switching to a different action with a probability that is proportional to the regret for that action.
  - If every player plays according to it, then the history plays converge to the set of correlated equilibrium.

What is our purpose?

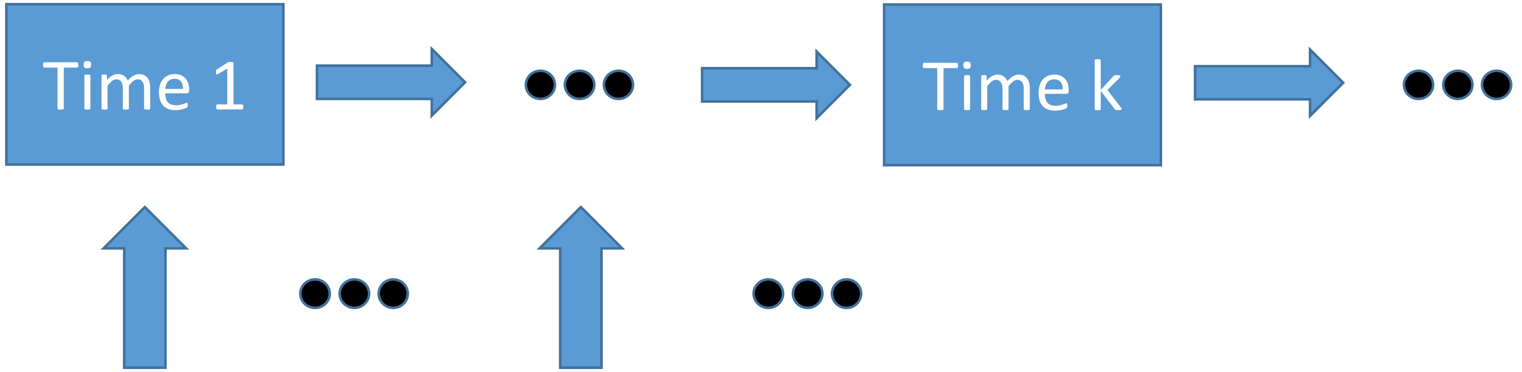
Game dynamics design

Nash equilibrium → Hard!

Correlated equilibrium → Regret based procedure!

A simple example of a regret-based procedure

Repeated game:
A same stage game is
played over and over again.



Stage Game

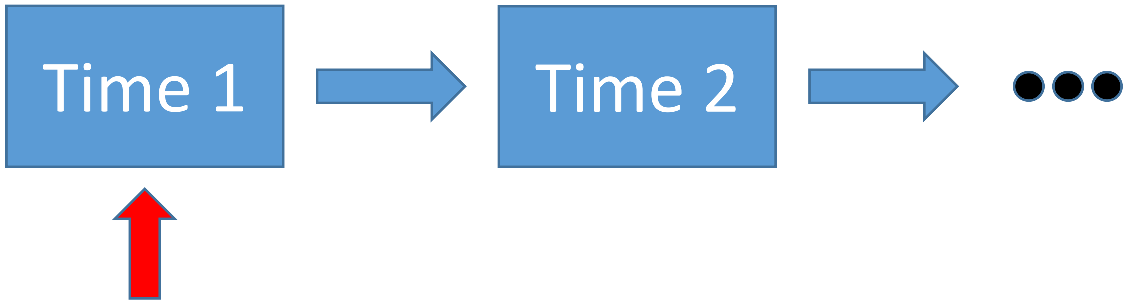
Players	2 clients
Actions	Send (S) or Hold (H)
Client i's traffic	t1 = t2 = 10
Server crash punishment	-c = -10

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0

Payoff table of a 2-client example

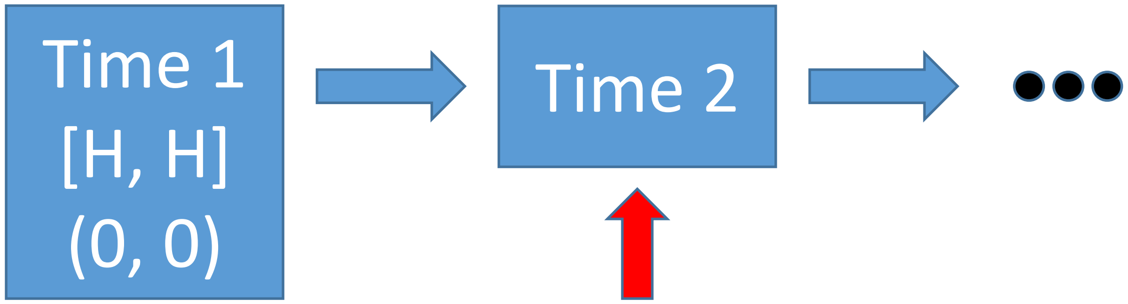
A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0



A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0



A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0

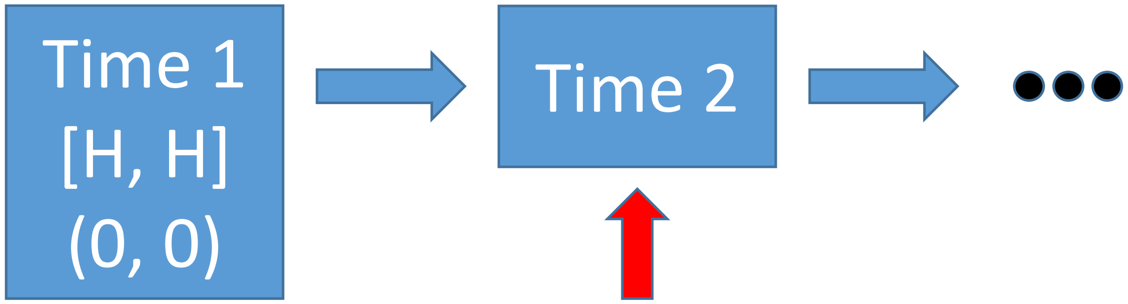
Time 1
[H, H]
(0, 0)
→ Time 2 → •••

For client 1:
Regret of not playing “S” = Profit([S, H]) – Profit([H, H]) = 10
For client 2:
Regret of not playing “S” = Profit([H, S]) – Profit([H, H]) = 10

For both client:
Prob(“S”) in the next move = (profit gain) / (normalize parameter) = 10/20 = 1/2
Prob(“H”) in the next move = 1 – 1/2 = 1/2

A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0



S -> H	N/A
H -> S	1/2

Client 1’s regret-based strategy table

S -> H	N/A
H -> S	1/2

Client 2’s regret-based strategy table

A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0

Time 1
[H, H]
(0, 0)

→

Time 2
[S, H]
(10, 0)

→ …

A simple example of a regret-based procedure

	S	H
S	-10, -10	10, 0
H	0, 10	0, 0

Time 1
[H, H]
(0, 0)

→

Time 2
[S, H]
(10, 0)

→ ●●●

↑ ●●●

Calculate regrets

↓

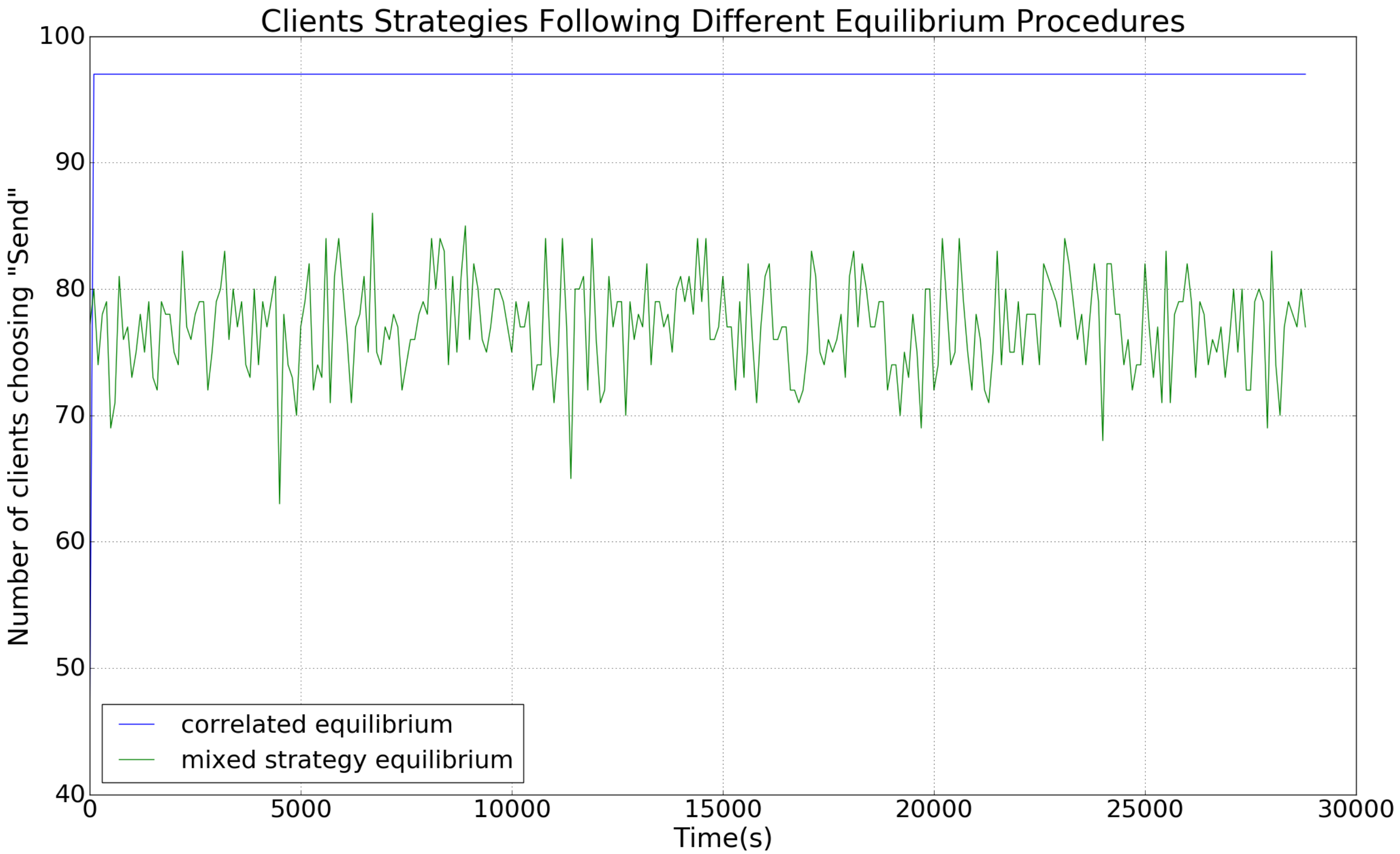
Calculate probability of strategy switching → Update strategy table → Make moves based on the table

The history distribution of clients' moves will converge to the set of correlated equilibrium of the game.

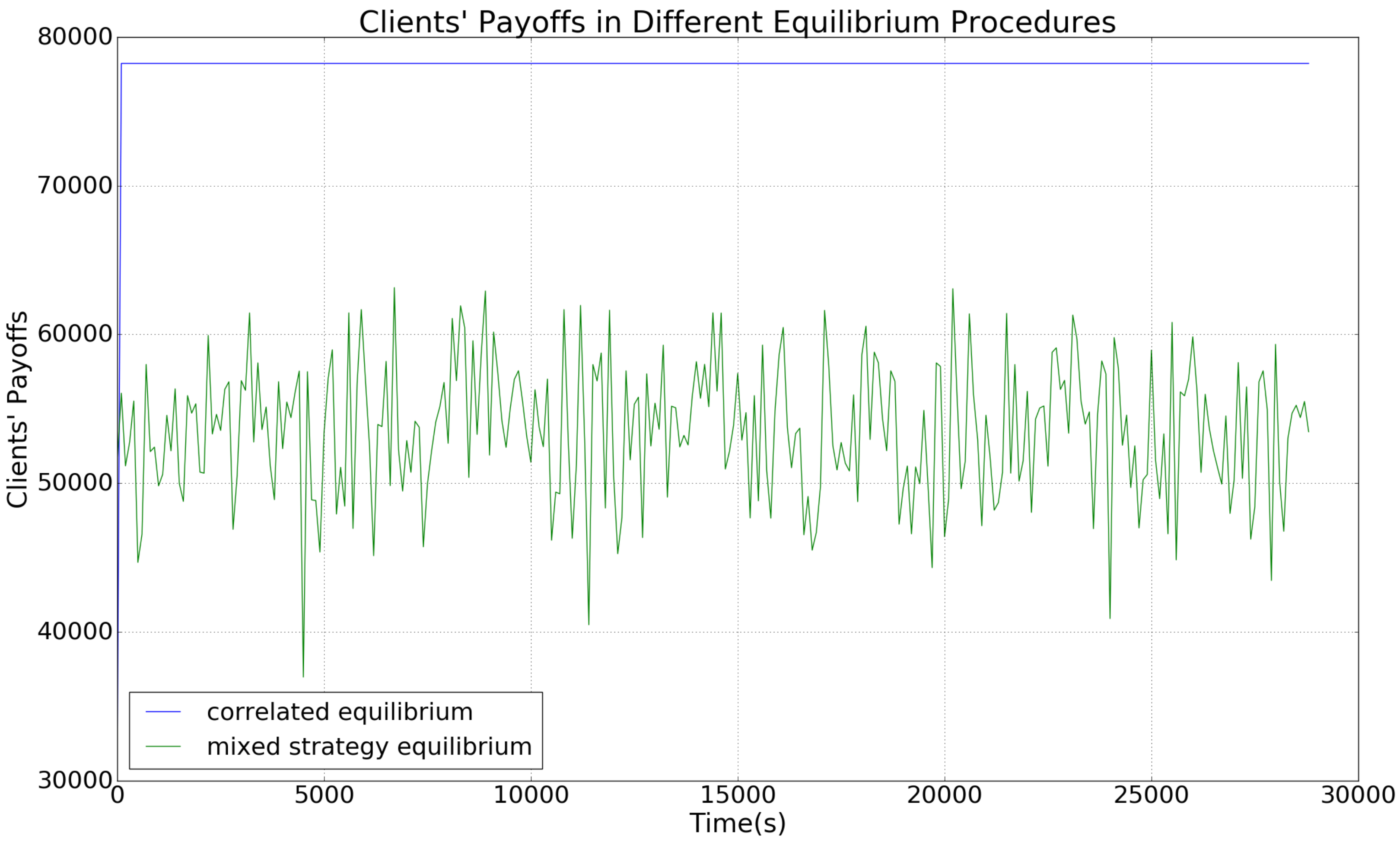
Simulation Settings

Parameter	Value
Number of clients n	98
Action set $\boldsymbol{S_i}$ of every client i	$\{S, H\}$
Amount of traffic client i possesses	$t_i \in [1, 1600]$
Punishment of server crash c	800
The big number μ	2401
Time unit (in second)	1
Simulation length (in second)	28800

Simulation Results – Clients' Behaviors



Simulation Results – Clients' Payoffs



What is next?

Refine the procedure

So that:

- Every client has a chance to send requests.
- It has behavior convergence, not history convergence, to the game's correlated equilibrium.
- It incorporates incomplete information factors – Bayesian game setting.

Conclusions

- The control system is the primary part in the whole accelerator suit. It assures the normal operations of the accelerators.
- This work aims to improve the control system's performance from the following two points of view:
  - Through simulations, develop more flexible and powerful tools to help testing and developing the control system.
  - Through theoretical analysis, improve understanding of the control system, and assisting the simulation work.

Thank You!