

# Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms

---

A. Takefusa<sup>\*1</sup>, S. Matsuoka<sup>\*2</sup>,  
H. Nakada<sup>\*3</sup>, K. Aida<sup>\*2</sup>, U. Nagashima<sup>\*4</sup>

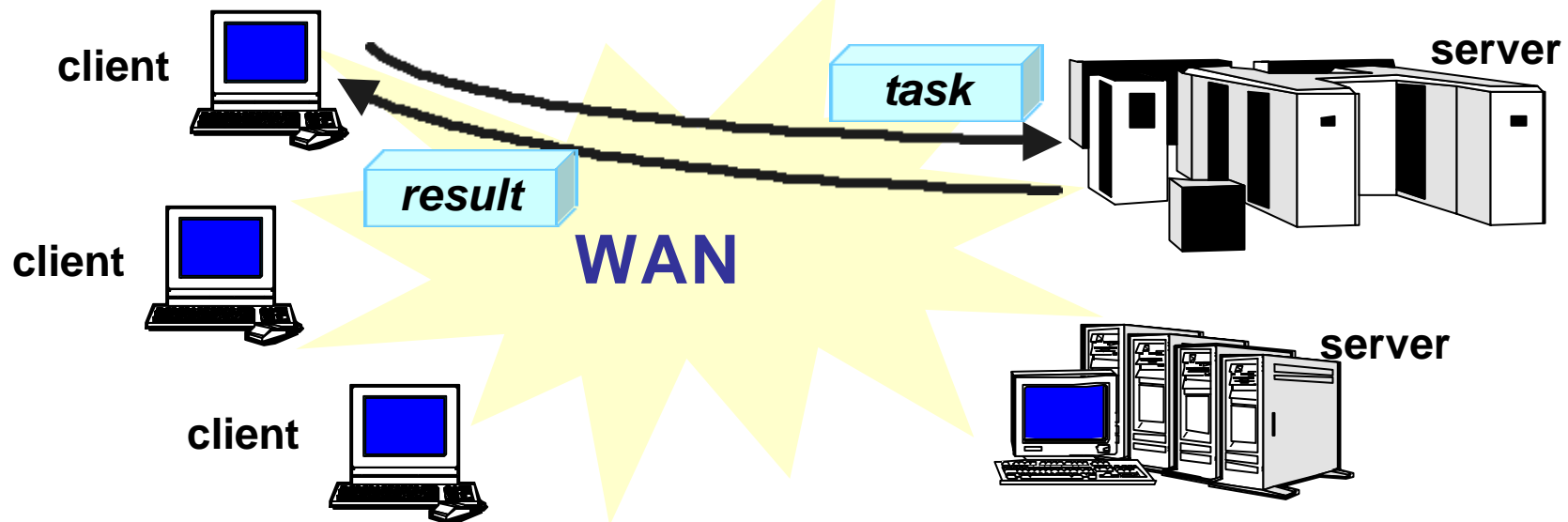
<sup>\*1</sup>Ochanomizu Univ., <sup>\*2</sup>TITECH, <sup>\*3</sup>ETL, <sup>\*4</sup>NIMC

<http://ninf.is.titech.ac.jp/bricks/>

# Global Computing System

- High performance global computing using computational and data resources
- Effective use of resources
  - Resource monitoring and prediction
  - Task scheduling

→ Scheduling Framework





# Evaluation of Scheduling in Global Computing

---

- Unrealistic to compare scheduling algorithms w/physical benchmarks
  - Reproducible large scale benchmarks are too difficult under various
    - Networks - topology, bandwidth, congestion, variance
    - Servers - architecture, performance, load, variance
- Validity of scheduling framework modules have not been well-investigated.
  - Benchmarking cost of monitor / predictor under real environment *HIGH*



# A Performance Evaluation System: *Bricks*

---

- Performance evaluation system for
  - Global computing scheduling algorithms
  - Scheduling framework components (e.g., sensors, predictors)
- Bricks provides
  - Reproducible and controlled evaluation environments
  - Flexible setups of simulation environments
  - Evaluation environment for existing global computing components (e.g., NWS)



# Outline

---

- Overview of the Bricks system
- The Bricks architecture
- Incorporating existing global computing components (ex. NWS)
- Bricks experiments

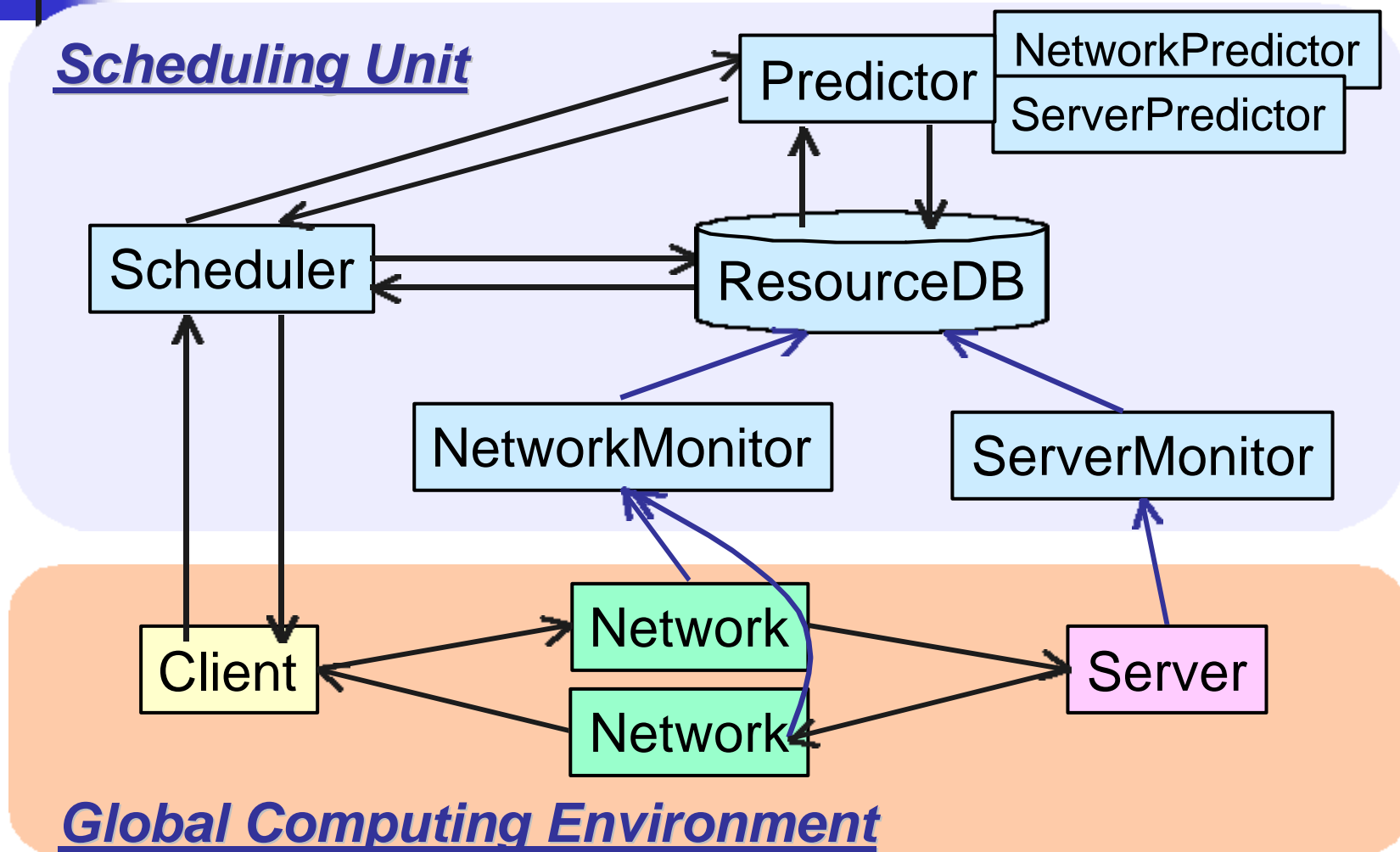


# Overview of Bricks

---

- Consists of simulated *Global Computing Environment* and *Scheduling Unit*.
- Allows simulation of various behaviors of
  - resource scheduling algorithms
  - programming modules for scheduling
  - network topology of clients and servers
  - processing schemes for networks and servers (various queuing schemes)using the *Bricks script*.
- Makes benchmarks of existing global scheduling components available

# The Bricks Architecture



# Global Computing Environment

- Client

- represents user of global computing system
- invokes global computing Tasks

Amount of data transmitted to/from server,  
# of executed instructions

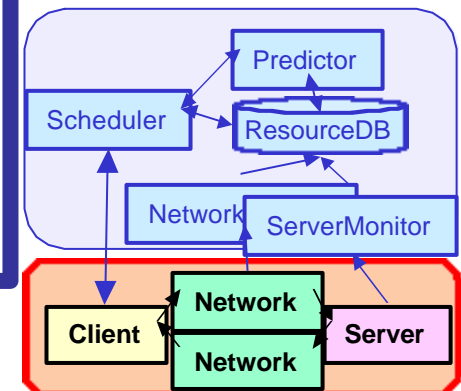
- Server

- represents computational resources

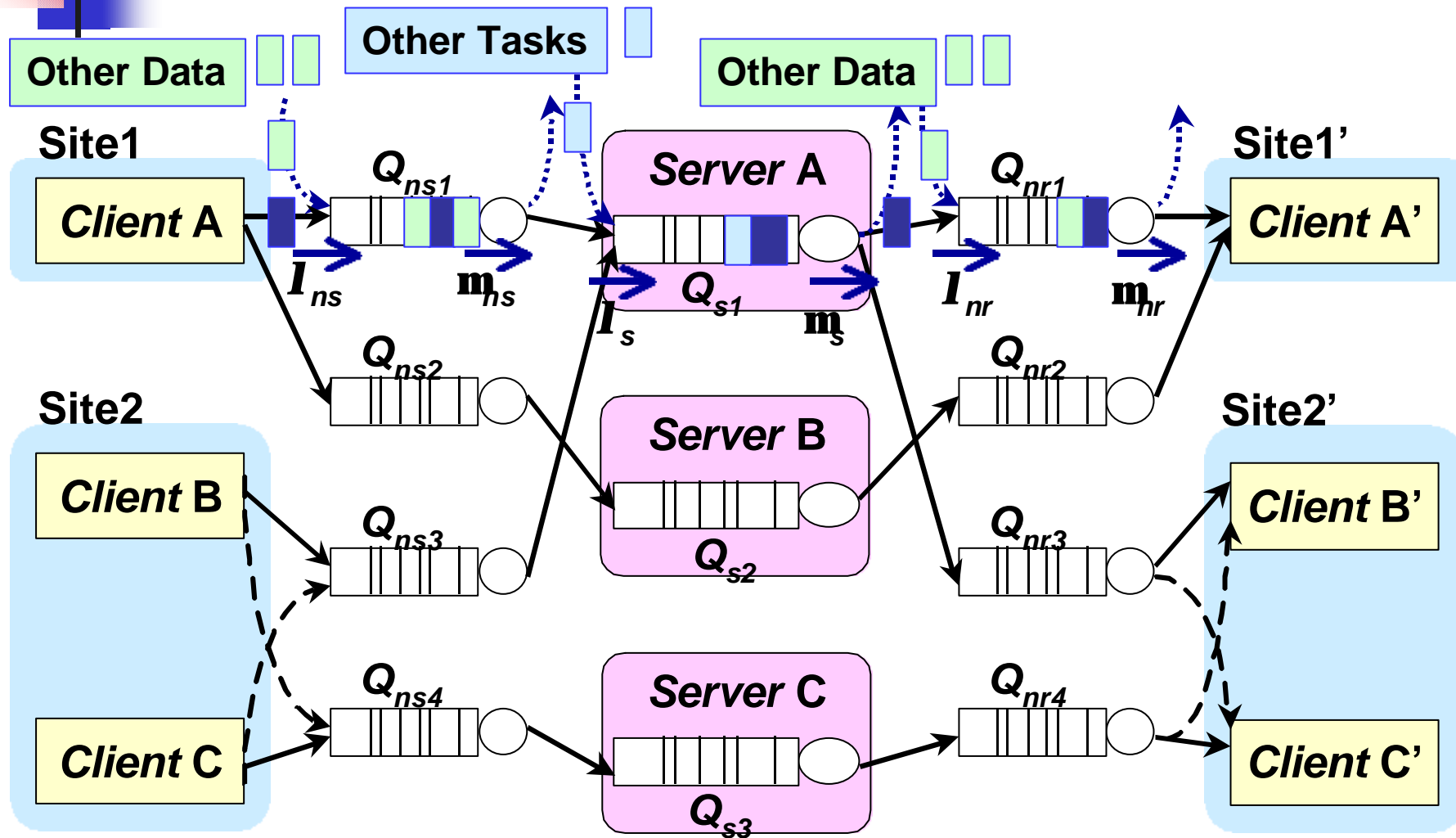
- Network

- represents the network interconnecting the Client and the Server

Represented using queues



# Queuing Model-based Global Computing Simulation [NASA Workshop 98]





# Communication/Server Models using queues in Bricks

---

- Extraneous data/jobs generation

Congestion represented by adjusting the amount of arrival data/jobs from other nodes/users [NASA Workshop98]

- Need to specify only several parameters

- ✗ Greater accuracy requires larger simulation cost

- Observed parameters

Bandwidth/performance at each step = observed parameters of real environment.

- Network/Server behaves as if real network/server

- Simulation cost lower than the previous model

- ✗ Need to accumulate the measurements

# Scheduling Unit

- NetworkMonitor/ServerMonitor

measures/monitors network/server status in global computing environments

- ResourceDB

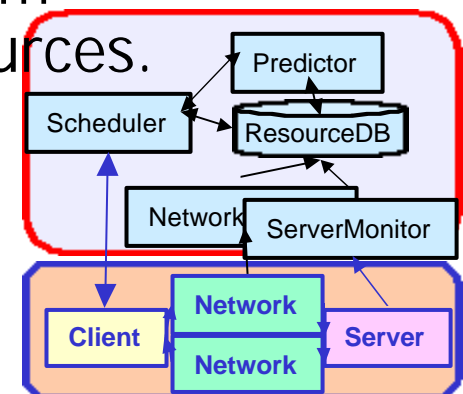
serves as scheduling-specific database, storing the values of various measurements.

- Predictor

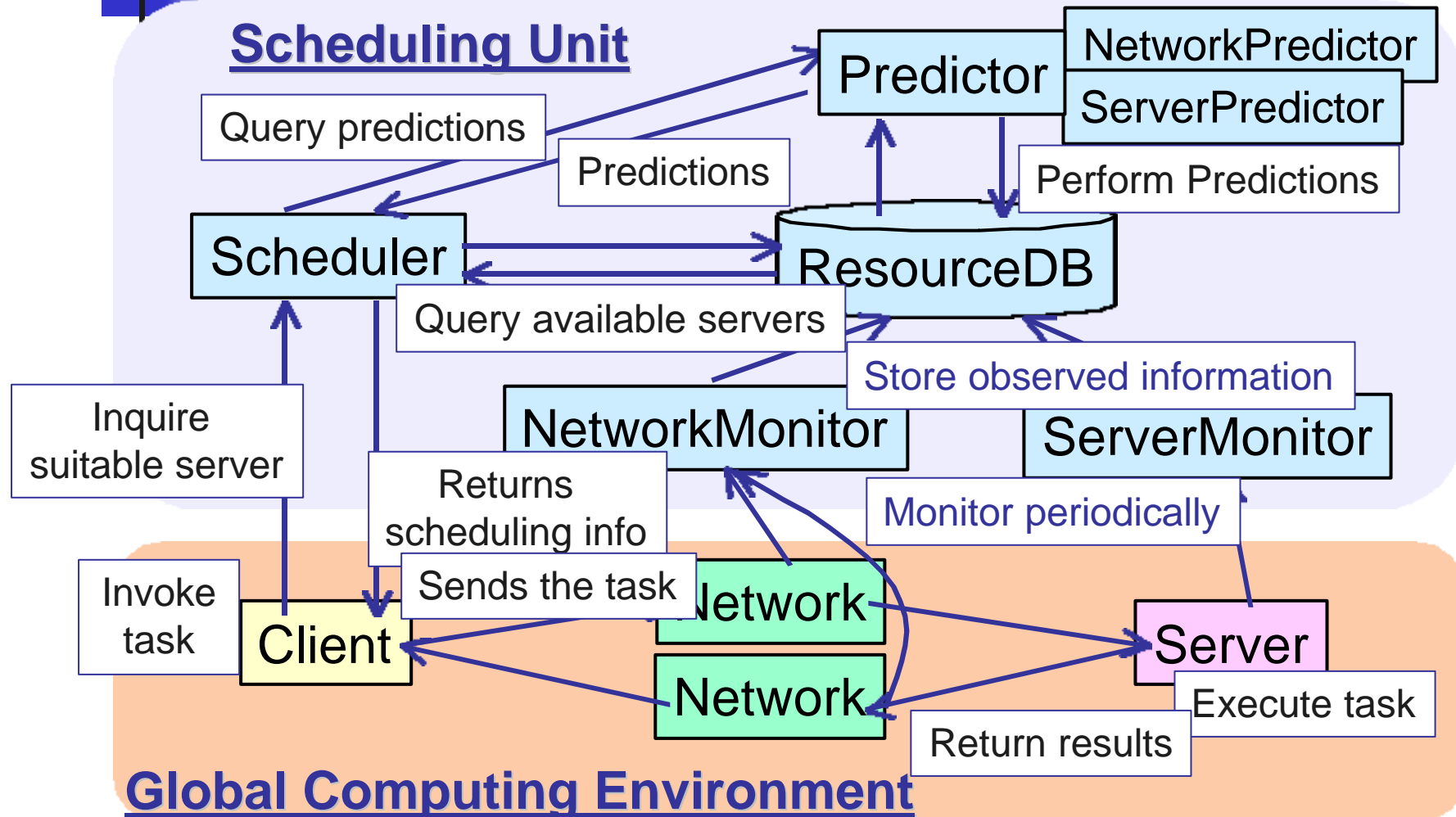
reads the measured resource information from ResourceDB, and predicts availability of resources.

- Scheduler

allocates a new task invoked by a client on suitable server machine(s)



# Overview of Global Computing Simulation with Bricks





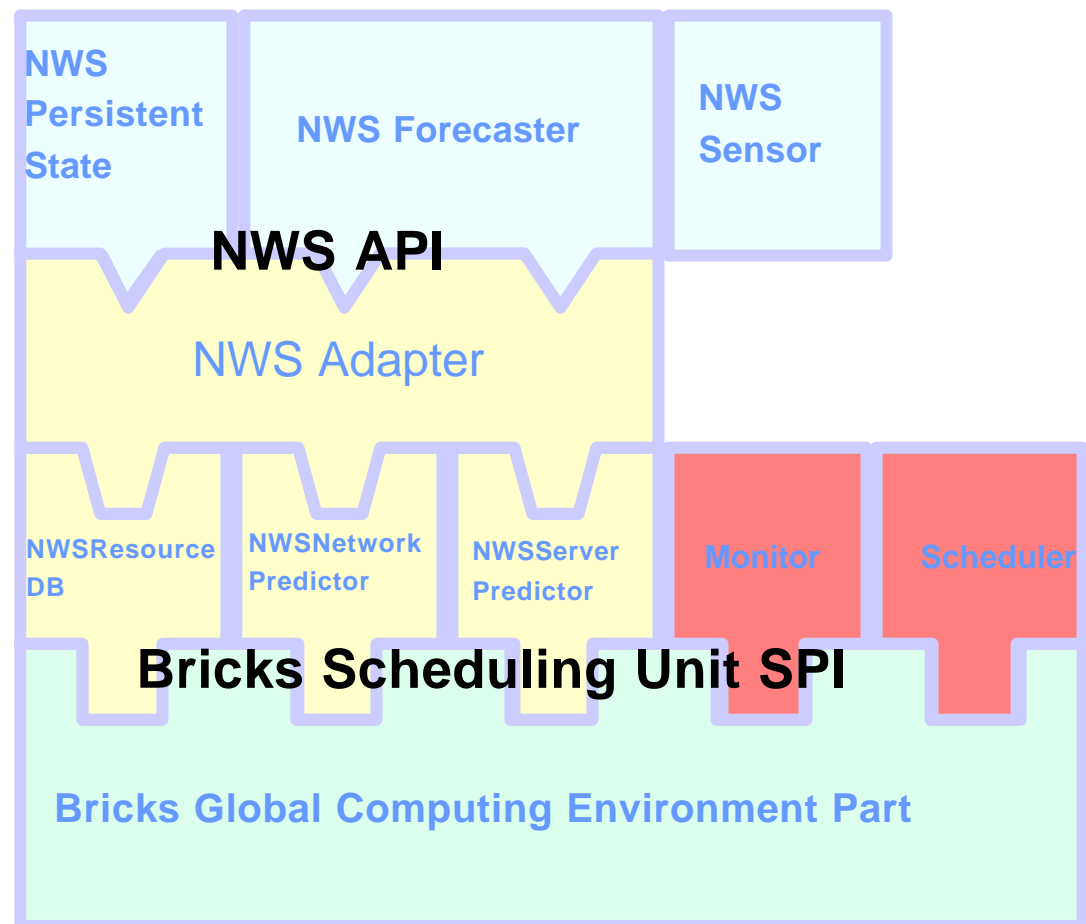
# Incorporating External Components

---

- Scheduling Unit module *replacement*
  - Replaceable with other Java scheduling components
  - Components could be external --- in particular, real global computing scheduling components
    - allowing their validation and benchmarking under simulated and reproducible environments
- Bricks provides **the Scheduling Unit SPI**.

# Scheduling Unit SPI

```
interface ResourceDB {  
    void putNetworkInfo();  
    void putServerInfo();  
    NetworkInfo getNetworkInfo();  
    ServerInfo getServerInfo();  
}  
interface NetworkPredictor {  
    NetworkInfo getNetworkInfo();  
}  
interface ServerPredictor {  
    ServerInfo getServerInfo();  
}  
interface Scheduler {  
    ServerAggregate selectServers();  
}
```





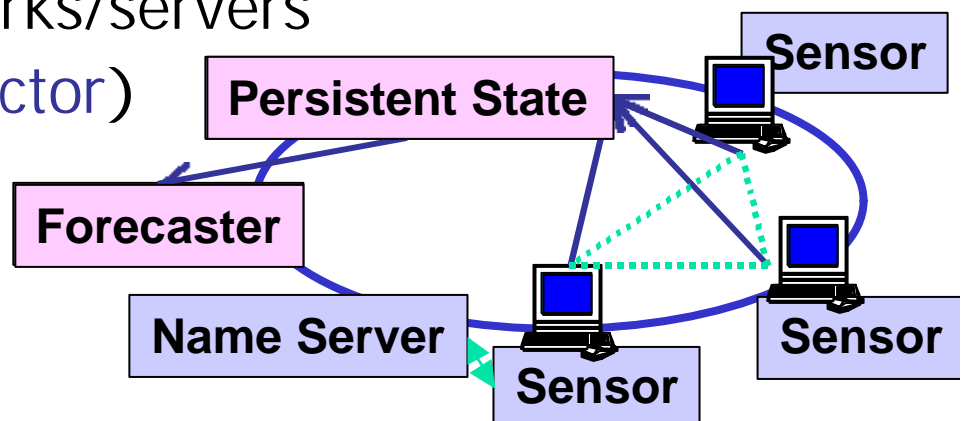
# Case study

---

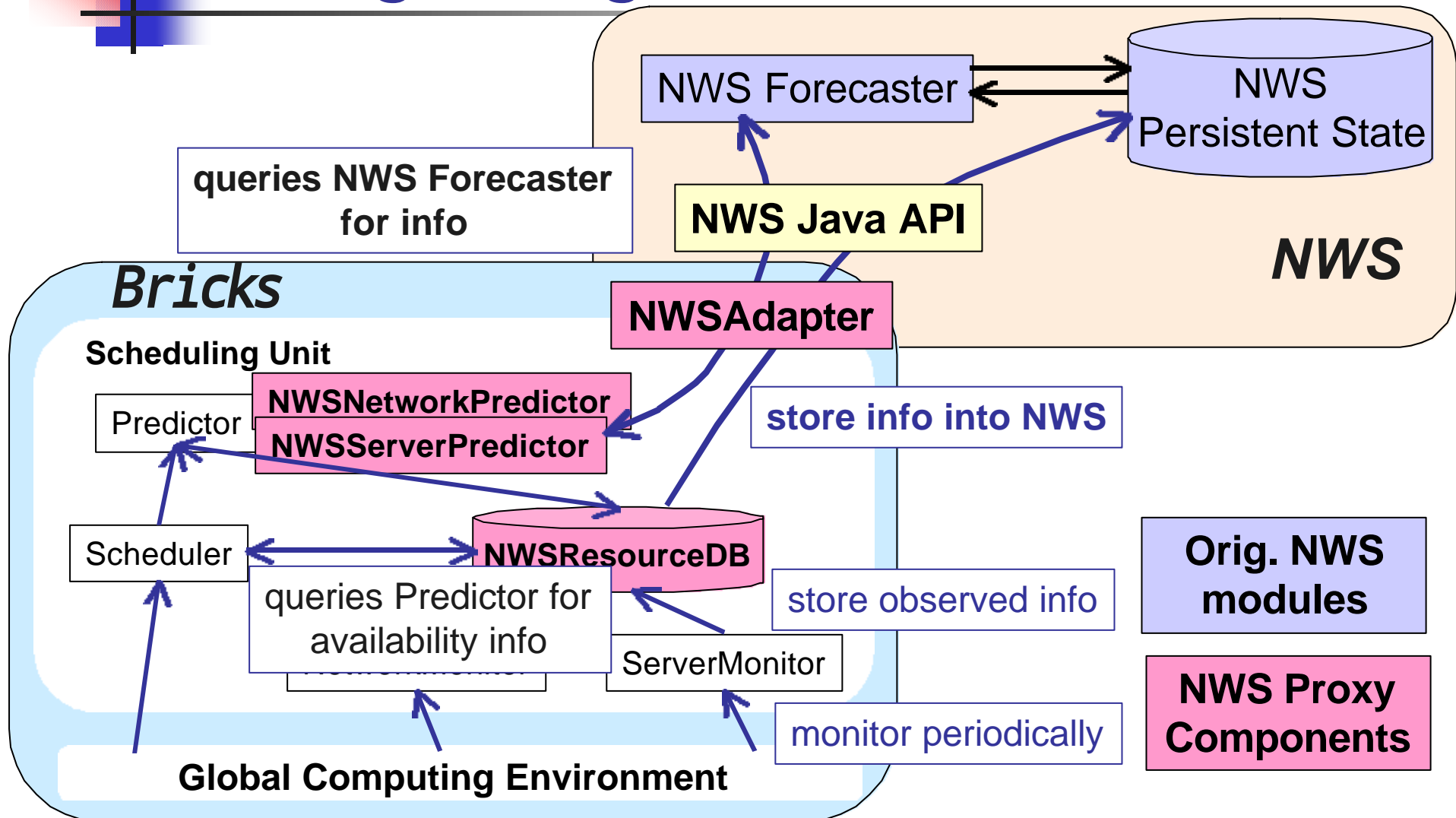
- NWS[UCSD] integration into Bricks
  - monitors and predicts the behavior of global computing resources
  - has been integrated into several systems, such as AppLeS, Globus, Legion, Ninf
  - Orig. C-based API
    - NWS Java API development
    - NWS run under Bricks

# The NWS Architecture

- **Persistent State** (→Replace ResourceDB)  
is storage for measurements
- **Name Server**  
manages the correspondence between the IP/domain address for each independently-running modules of NWS
- **Sensor** (→Network/ServerMonitor)  
monitors the states of networks/servers
- **Forecaster** (→ Replace Predictor)  
predicts availability of the resources



# Integrating NWS into Bricks





# Bricks Experiments

---

■ The experiments conducted by running NWS under a real environment vs. Bricks environment

Whether can Bricks provide

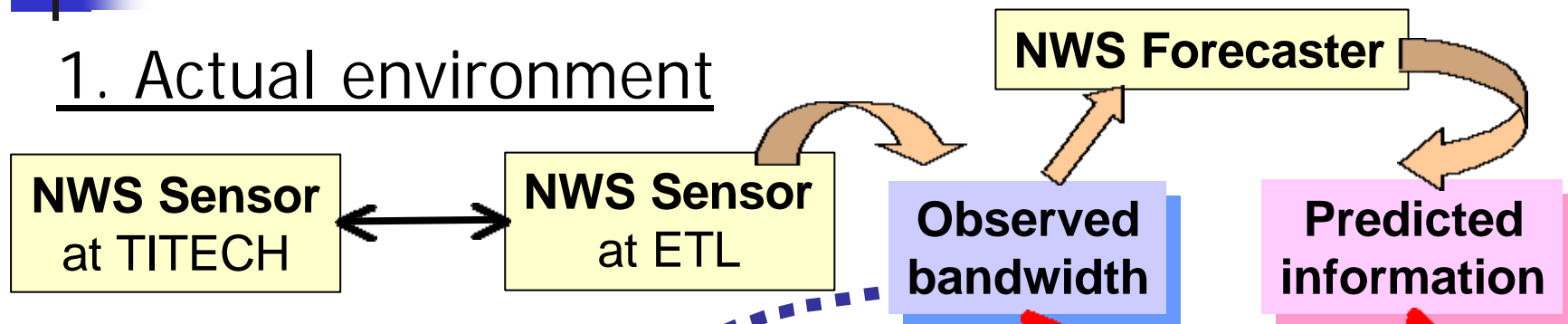
- A simulation environment for global computing with reproducible results?
- A benchmarking environment for existing global computing components?

■ Experiment Procedure

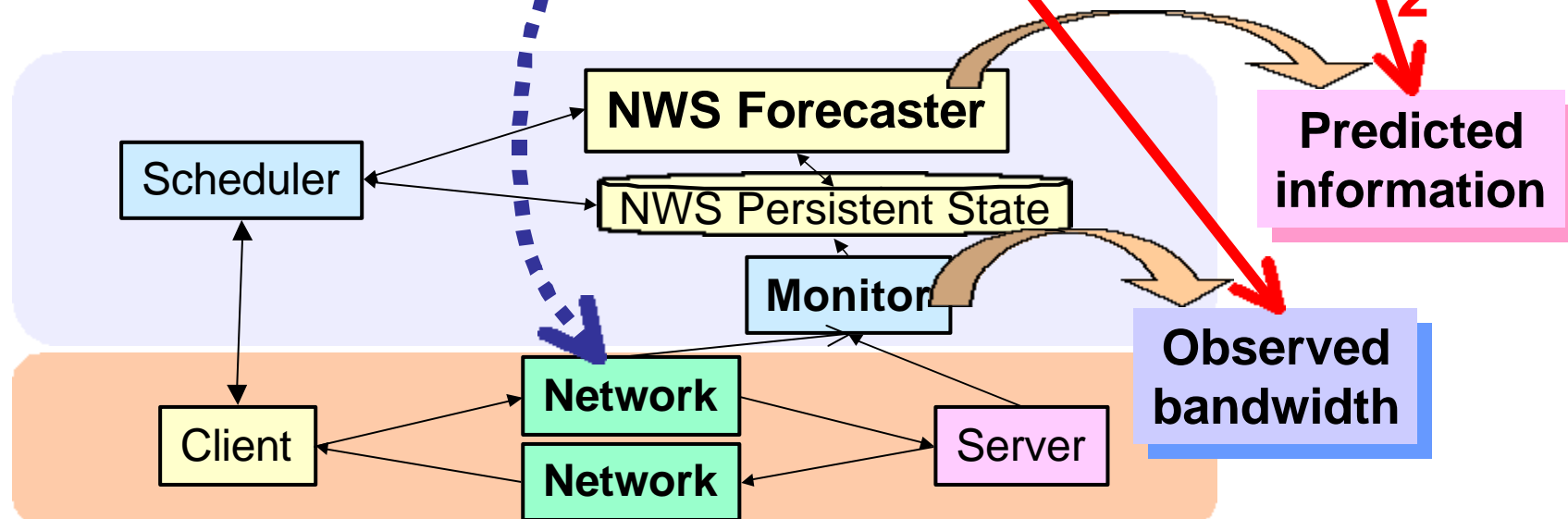
1. Actual measurement: Have NWS modules measure and predict parameters real wide-area environment
2. Simulation: Drive Bricks under the NWS measurements, and have NWS Forecaster make predictions under Bricks simulation

# Overview of Experiments

## 1. Actual environment

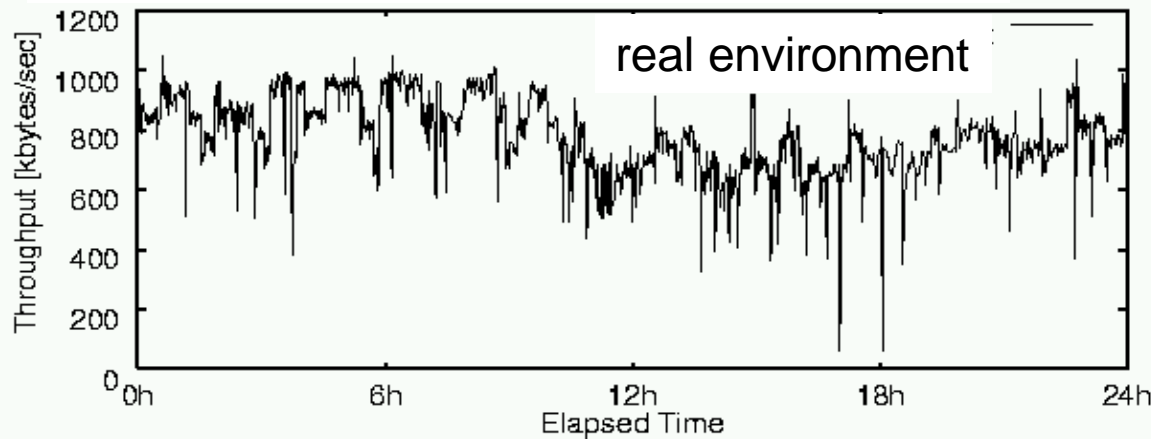


## 2. Bricks simulated environment



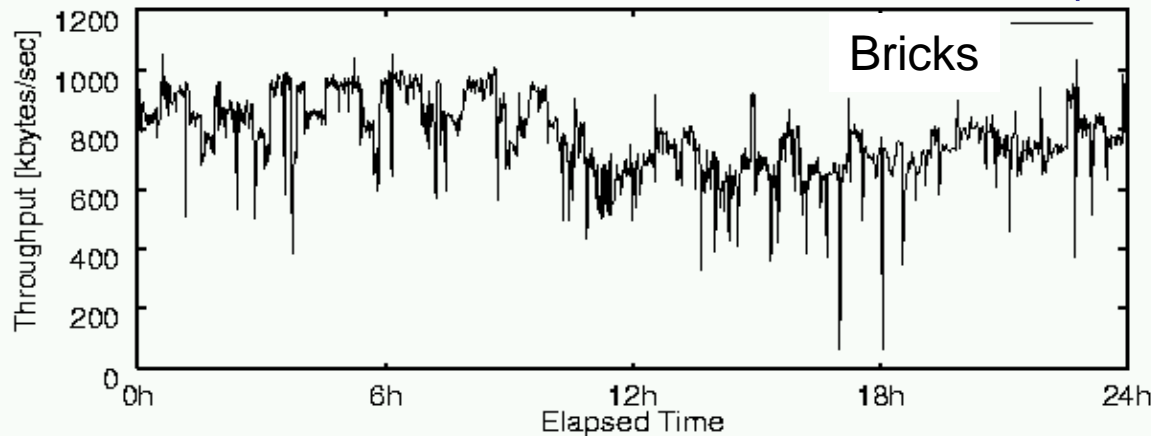
# Bricks Experimental Results 1: Comparison of Observed Bandwidth

## Under real environment(24hours)



- The Bandwidth measured under Bricks is quite similar to that for the real environment

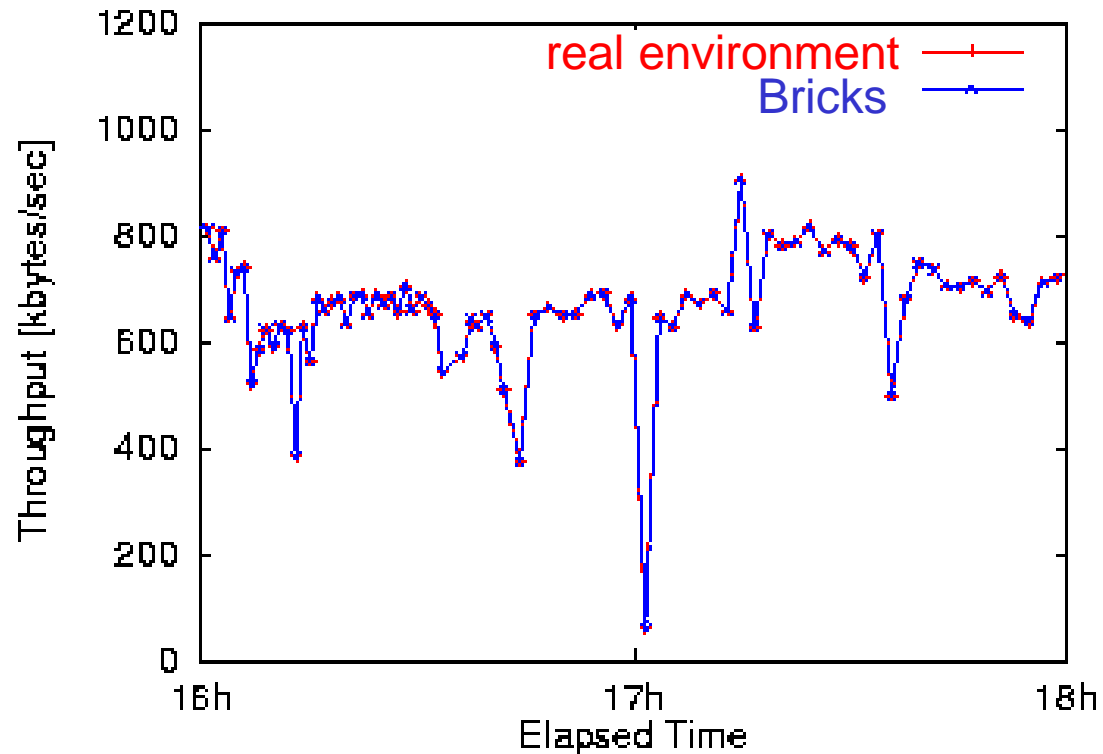
## Under Bricks simulated environment(24hours)



NWS:  
TITECH  $\leftrightarrow$  ETL  
network monitoring: 60[sec]  
network probe :300[KB]  
Bricks:  
cubic spline interpolation

# Bricks Experimental Results 1: Comparison of Observed Bandwidth

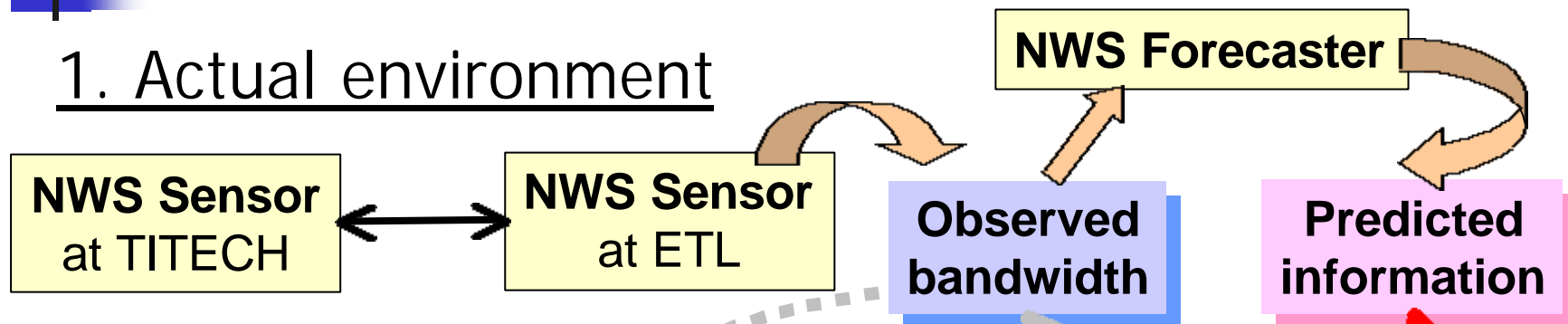
2 hours comparison



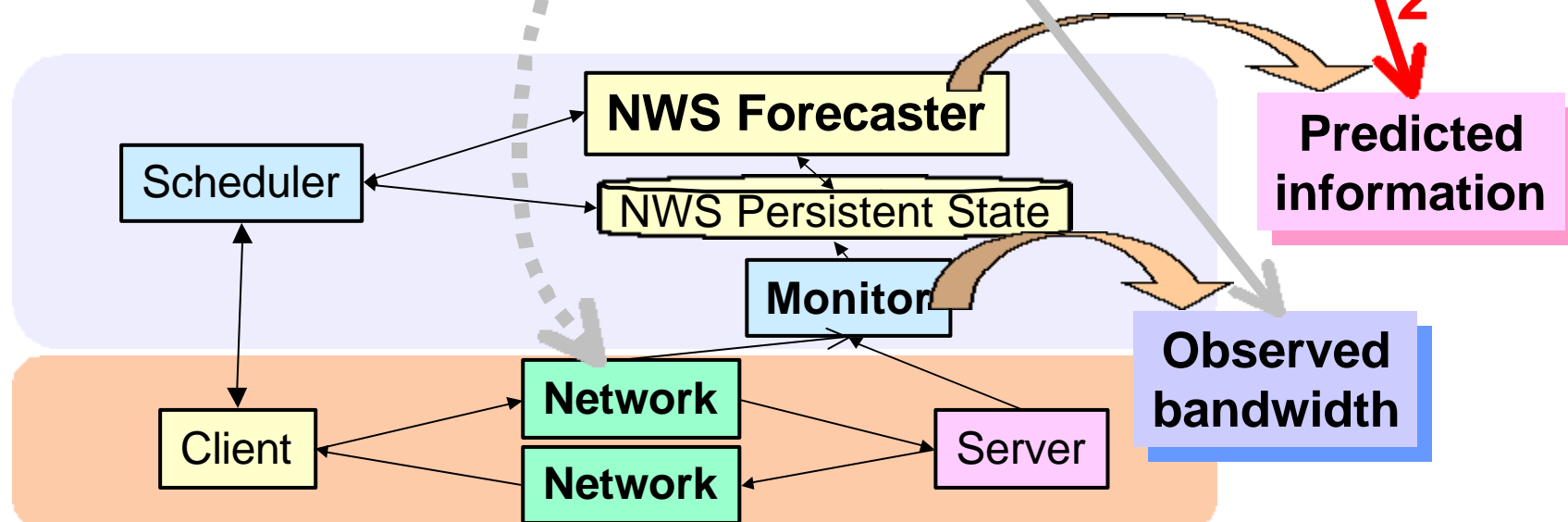
- Bandwidths measured under real environment and Bricks coincide well

# Overview of Experiments

## 1. Actual environment

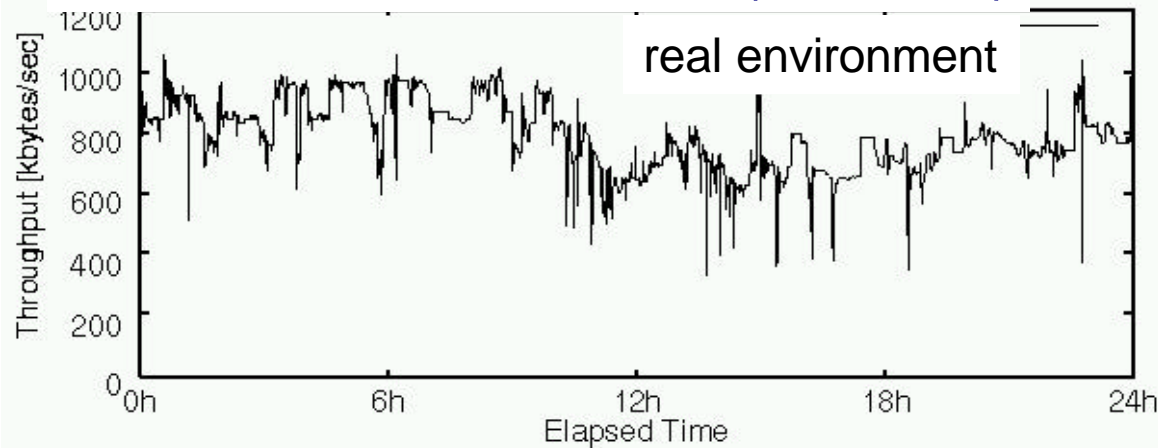


## 2. Bricks simulated environment



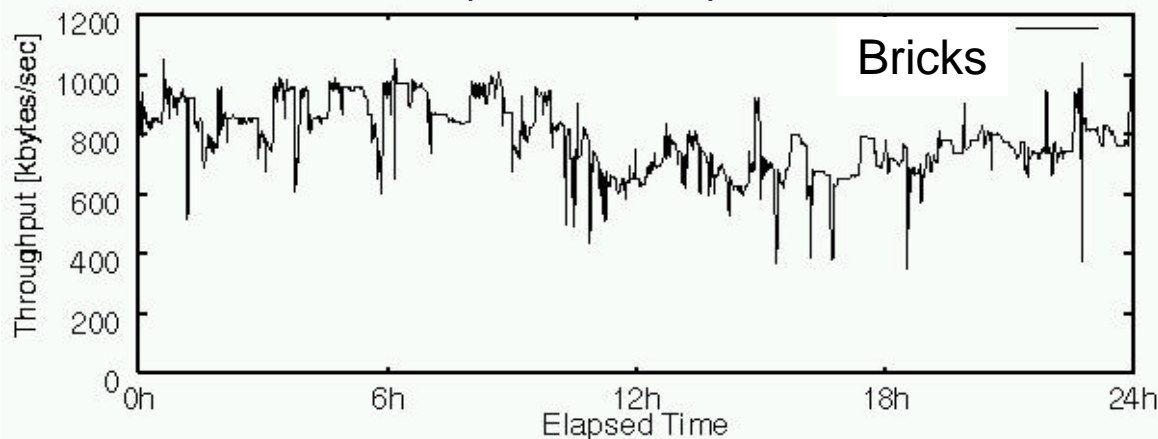
# Bricks Experimental Results 2: Comparison of Predicted Bandwidth

## Under real environment(24hours)



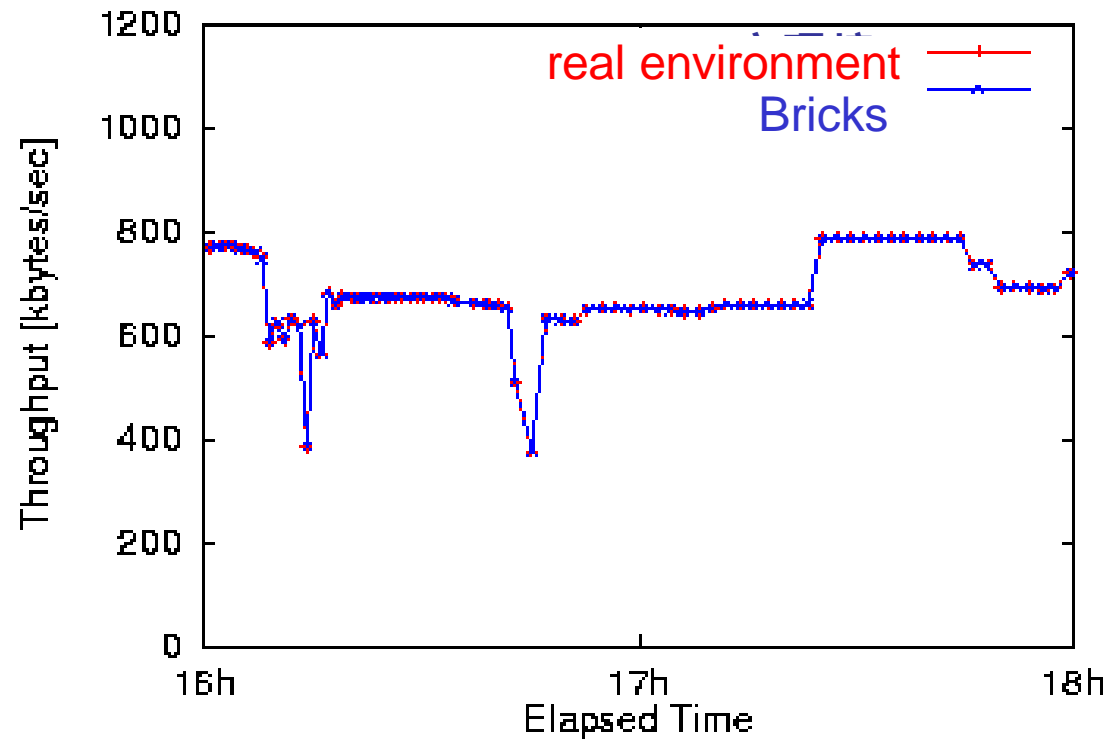
- The NWS Forecaster functions and behaves normally under Bricks

## Under Bricks (24hours)



# Bricks Experimental Results 2: Comparison of Predicted Bandwidth

2 hours comparison  
of predicted bandwidth  
by the NWS Forecaster



- The both predictions are very similar.
- ® Bricks provides existing global computing components with a benchmarking environment



# Related Work

---

- Osculant Simulator [Univ. of Florida]
  - evaluates Osculant: bottom-up scheduler for heterogeneous computing environment
  - makes various simulation settings available
- WARMstones [Syracuse Univ.]
  - is similar to Bricks, although it seems not have been implemented yet?
  - provides an interface language (MIL) and libraries based on the MESSIAHS system to represent various scheduling algorithms → Bricks provides SPI
  - has not provided a benchmarking environment for existing global computing components



# Conclusions

---

- We proposed the **Bricks** performance evaluation system for global computing scheduling
  - Bricks provides multiple simulated reproducible benchmarking environments for
    - Scheduling algorithms
    - Existing global computing components
- Bricks experiments showed
  - Bricks could perform accurate simulation
  - The NWS Forecaster behaved normally under Bricks

→ Evaluation of existing global computing components now possible



# Future Work

---

- Simulation model needs to be more sophisticated and robust
  - Task model for parallel application tasks
  - Server model for various server machine architectures(e.g., SMP , MPP) and scheduling schemes(e.g., LSF)
- Component integration (e.g., direct support for IP)
- Investigation of various scheduling algorithms
  - On parallel simulation cluster

# Cluster



- 64PE cluster at Matsuoka Lab., TITECH
  - Pentium II 350MHz
  - Memory: 128MB
  - NIC: Intel EtherExpress Pro 10/100
- Parallel simulations of global computing environments



# Acknowledgments

---

- The NWS team
  - Rich Wolski [UTK]
  - Jim Hayes, Fran Berman [UCSD]
- The Ninf team
  - Satoshi Sekiguchi [ETL]
  - Mitsuhisa Sato [RWCP]
  - Many others [ETL, TITECH]