

# UDT: UDP based Data Transfer

Protocol, Results, and Implementation Experiences

Yunhong Gu & Robert Grossman

Laboratory for Advanced Computing / Univ. of Illinois at Chicago

Bill Allcock & Raj Kettimuthu

Globus Alliance / Argonne National Laboratory

## Outline

1. UDT Protocol Introduction
2. UDT Congestion Control (Yunhong Gu)
3. Simulation Studies
4. Implementation Experiences at ANL (Raj Kettimuthu)

# 1. UDT Protocol Introduction

02/17/2004

PFLDnet 2004

3

## Design Goals and Target Applications

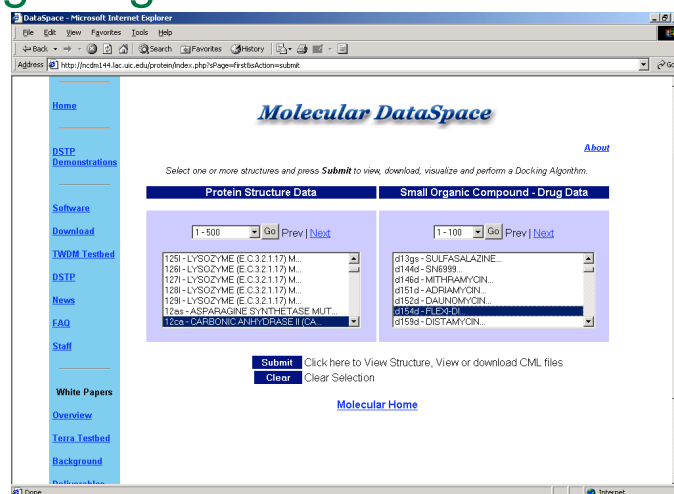
- Target applications
  - Data intensive applications requiring one to dozens of high volume flows
- Goals: Easy, Fast, Fair, Friendly
  - Easy to deploy today: application layer
  - High utilization of available bandwidth for single or multiple high volume flows
  - Intra-protocol fairness
  - RTT independence
  - TCP friendly

02/17/2004

PFLDnet 2004

4

## Data Webs and Data Grids: Example Integrating Bioinformatics Data

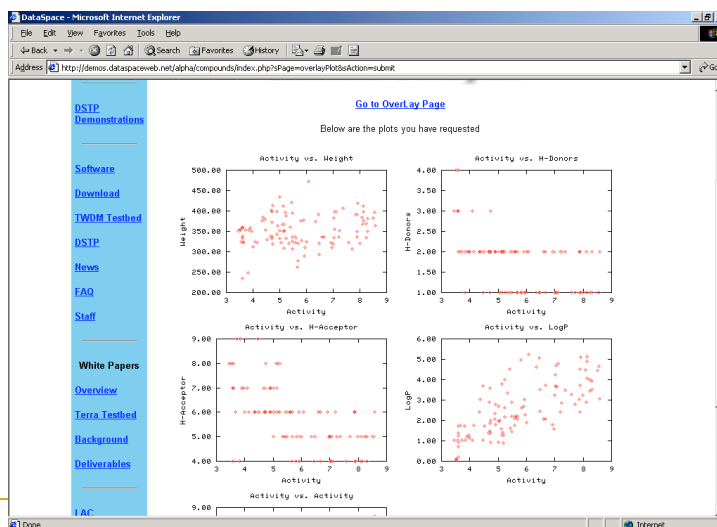


02/17/2004

PFLDnet 2004

5

## Typical Query: Integrate Data and Look for Correlations



02/17/2004

PFLDnet 2004

6

## What's UDT?

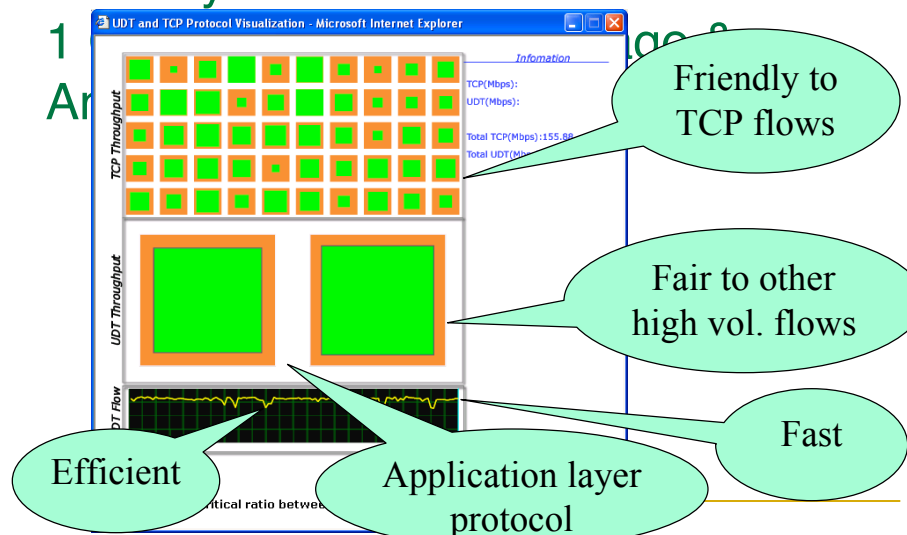
- UDT: UDP based Data Transfer
  - Reliable, application level, duplex, transport protocol, over UDP with reliability, congestion, and flow control
  - Successor to SABUL
  - Open source C++ library on Source Forge
- Two orthogonal parts
  - The UDT protocol framework that can be implemented above UDP, with any suitable congestion control algorithms
  - The UDT congestion control algorithm, which can be implemented in any transport protocols such as TCP

02/17/2004

PFLDnet 2004

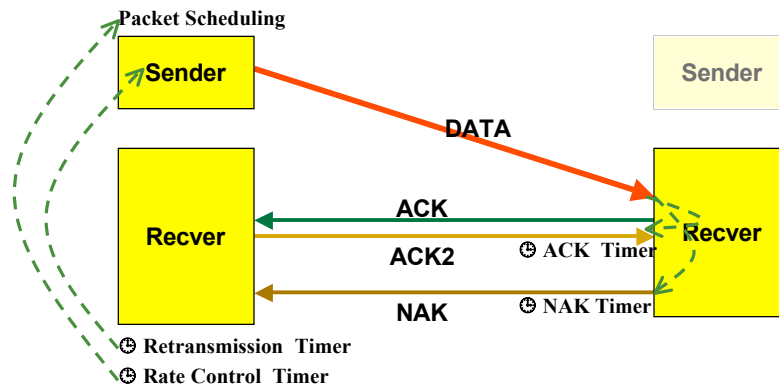
7

## SC 03 BWC: UDT is Fast, Fair & Friendly



8

## UDT Protocol Framework



02/17/2004

PFLDnet 2004

9

## UDT Protocol

- Packet based sequencing
- ACK sub-sequencing
- Explicit loss information feedback (NAK)
- Four timers: rate control, ACK, NAK and retransmission timer
  - Rate control and ACK are triggered periodically
  - NAK timer is used to resend loss information if retransmission is not received in an increasing time interval

02/17/2004

PFLDnet 2004

10

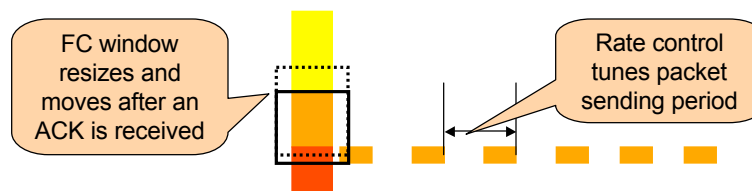
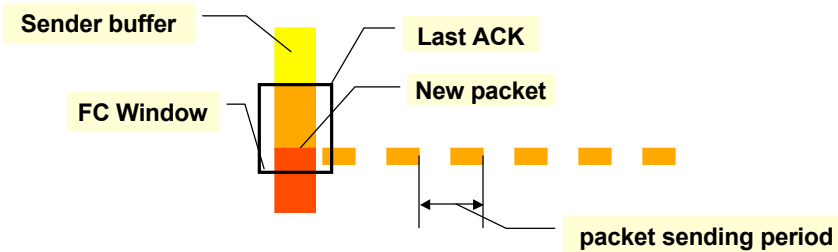
## 2. UDT Congestion Control

02/17/2004

PFLDnet 2004

11

### Congestion Control Framework



02/17/2004

PFLDnet 2004

12

## Congestion Control Framework


- Rate based congestion control (Rate Control)
  - RC tunes the packet sending period.
  - RC is triggered periodically.
  - RC period is constant of 0.01 seconds.
- Window based flow control (Flow Control)
  - FC limits the number of unacknowledged packets.
  - FC is triggered on each received ACK.
- Slow start is controlled by FC
  - Similar to TCP, but only occurs at the session beginning.

02/17/2004

PFLDnet 2004

13

## Rate Control

- AIMD: Increase parameter is related to link capacity and current sending rate; Decrease factor is 1/9, but not decrease for all loss events.
- Link capacity is probed by packet pair, which is sampled UDT data packets.
  - Every 16th data packet and its successor packet are sent back to back to form a packet pair.
 
  - The receiver uses a median filter on the interval between the arrival times of each packet pair to estimate link capacity.

02/17/2004

PFLDnet 2004

14

## Rate Control Example

C = current sending rate

inc = increase parameter for rate control

B = estimated using packet, e.g. B = 10Gbps below  
MSS = packet size (here 1500 bytes)

C (Mbps)	B - C (Mbps)	inc (Pkts)
[0, 9000)	(1000, 10000]	10
[9000, 9900)	(100, 1000]	1
[9900, 9990)	(10, 100]	0.1
[9990, 9999)	(1, 10]	0.01
[9999, 9999.9)	(0.1, 1]	0.001
9999.9+	<0.1	0.00067

02/17/2004

PFLDnet 2004

15

## Rate Control – General Case

- Number of packets to be increased in next rate control period (RCTP) time is:  

$$inc = \max(10^{\lceil \log_{10}((B-C) \times MSS \times 8) \rceil} \times \beta / MSS, 1 / MSS)$$
 where  $B$  is estimated link capacity,  $C$  is current sending rate. Both are in packets per second. MSS is the packet size in bytes.  $\beta = 1.5 \times 10^{-6}$ .
- Decrease sending rate by 1/9 when a NAK is received, but only if
  1. largest lost sequence number in NAK is greater than the largest sequence number when last decrease occurred; or
  2. The number of NAKs since last decrease has exceeded a threshold, which increases exponentially and is reset when condition 1 is satisfied.

02/17/2004

PFLDnet 2004

16



## Flow Control

- $W = W * 0.875 + AS * (RTT + ATP) * 0.125$   
(window size)
- ATP is the ACK timer period, which is a constant of 0.01 seconds.
- AS is the packets arrival speed at receiver side.
  - The receiver records the packet arrival intervals. AS is calculated from the average of latest 16 intervals after a median filter.
- It is carried back within ACK.

02/17/2004

PFLDnet 2004

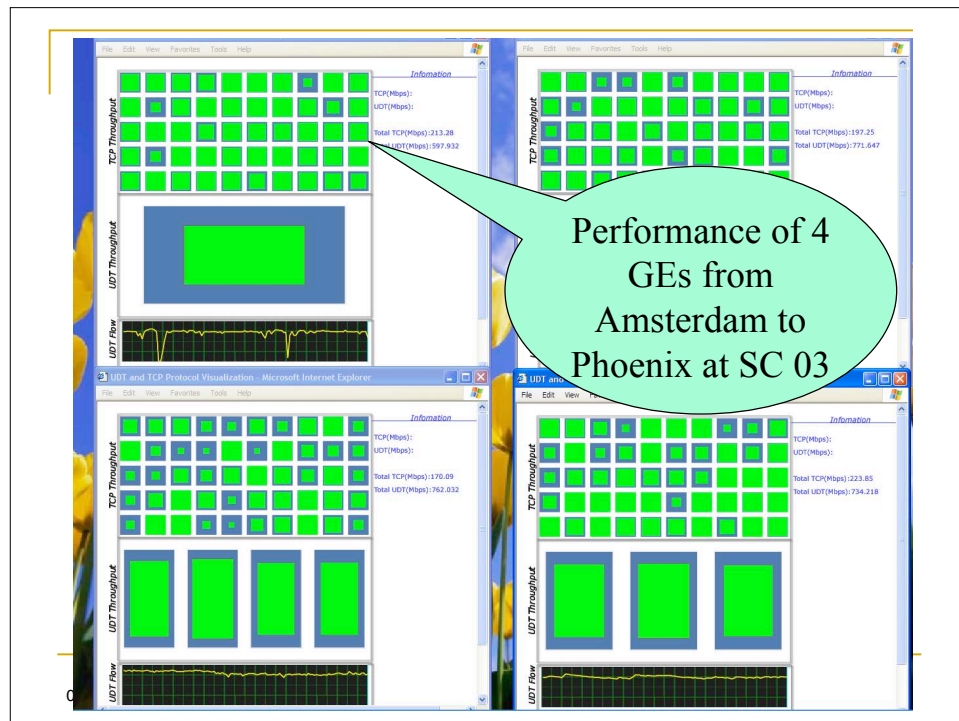
17

## 3. UDT Experimental Results

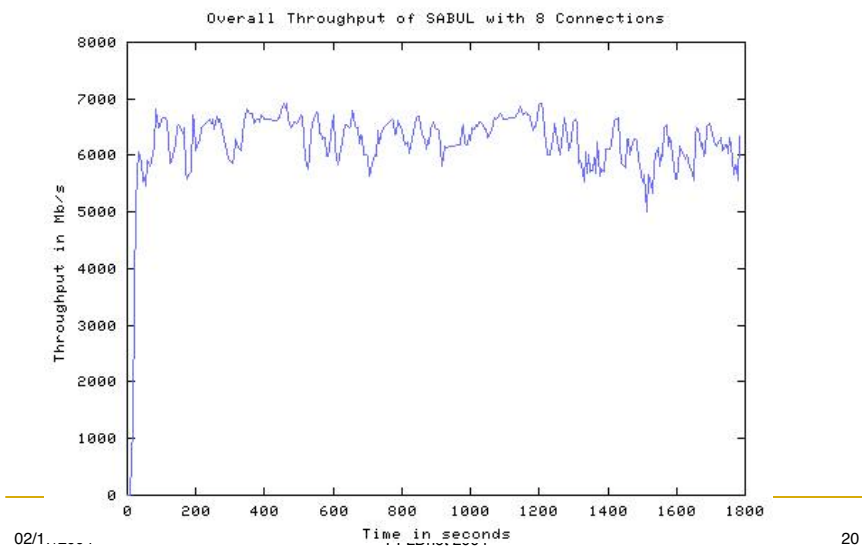
02/17/2004

PFLDnet 2004

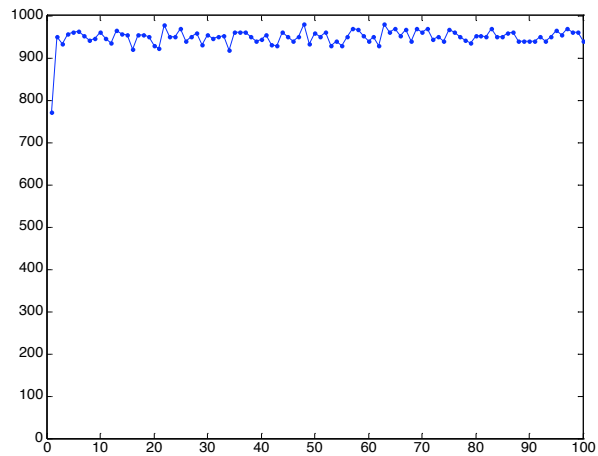
18



## 8 Aggregated Trans-Atlantic UDT Flows



## Implementation: Performance

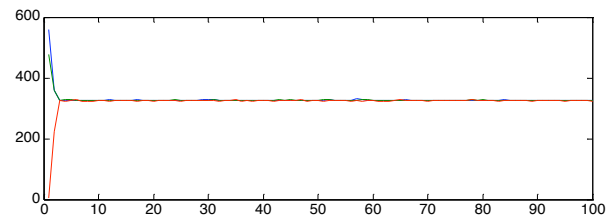


02/17/2004

PFLDnet 2004

21

## Implementation: Intra-protocol Fair

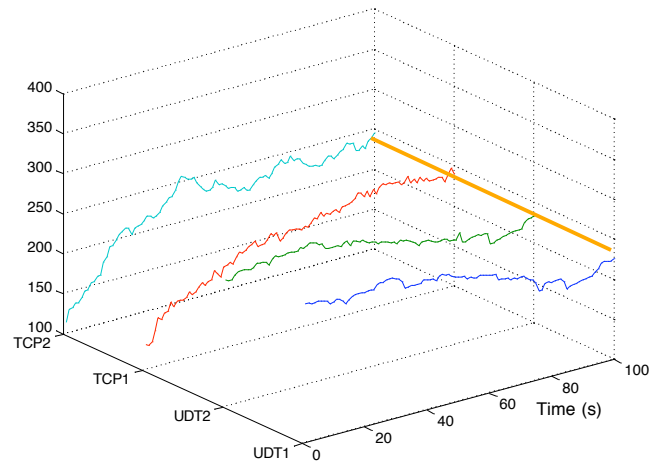


02/17/2004

PFLDnet 2004

22

## Implementation: TCP Friendliness



02/17/2004

PFLDnet 2004

23

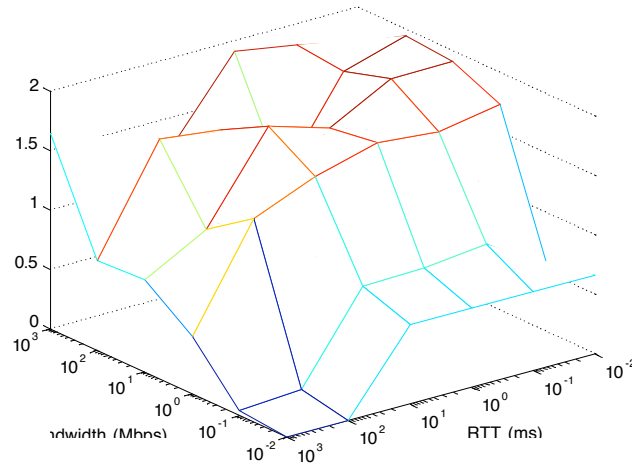
## 3. UDT Simulation Results

02/17/2004

PFLDnet 2004

24

## Simulation: TCP Friendliness

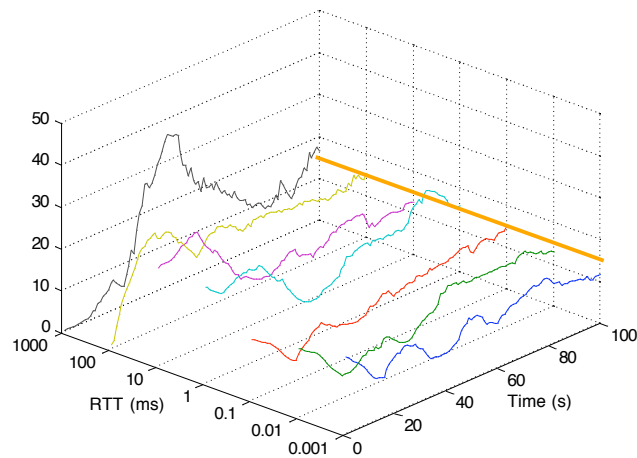


02/17/2004

PFLDnet 2004

25

## Simulation: RTT Independence

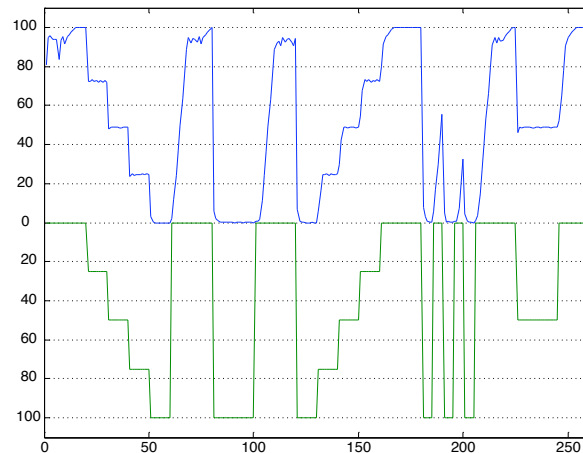


02/17/2004

PFLDnet 2004

26

## Simulation: Convergence/Stability



02/17/2004

PFLDnet 2004

27

## For More Information

- Technical Reports [www.ncdm.uic.edu](http://www.ncdm.uic.edu)
- Internet Draft: draft-gg-udt-xx.txt
- UDT source code  
[sourceforge.net/projects/dataspace](http://sourceforge.net/projects/dataspace)

02/17/2004

PFLDnet 2004

28

## 4. Implementation

### Experiences of UDT Driver for Globus XIO

Bill Allcock & Raj Kettimuthu  
Globus Alliance  
Argonne National Laboratory

## Globus XIO

- Extensible input/output library for the Globus toolkit®.
- Simple intuitive open/close/read/write API.
- Provides a driver development interface.
- Framework takes care of non-protocol ancillary requirements such as error handling etc.
- As Globus XIO has a built in UDP driver, the framework assists greatly in developing reliable layers on top of UDP.
- More details can be found at <http://www-unix.globus.org/developer/xio>

## Improvements Made to UDT

- To make UDT closely resemble TCP, developed server interface to handle multiple connection requests
- Server listens on a known port for receiving connection requests
- Upon receiving a request, a new socket created and the port information communicated to the client

02/17/2004

PFLDnet 2004

31

## Improvements Made to UDT (cont.)

- Client establishes a new connection to this port for data transfer
- Introduced some changes to the handshake mechanism
- Requirements that we had
  - Receiver not expected to know the transfer size.
  - Sender does not communicate the transfer size to the receiver.

02/17/2004

PFLDnet 2004

32



## Improvements Made to UDT (cont.)

- Completion of transfer intimated by closing UDT
- Had to introduce a close state machine into the protocol
- Included new control messages for close handling

02/17/2004

PFLDnet 2004

33

## Performance

- Initial results
  - Average throughput of 97 MBps on a GigE LAN
  - Average throughput of 33 MBps over the wide area link from ANL to LBL (bottleneck is OC12 link)
  - Throughput over the wide area link is low compared to the throughput achieved by the UIC implementation
  - Performance should improve with next implementation

02/17/2004

PFLDnet 2004

34

## Performance (cont.)

- Exploring the cause for the difference in performance
- Known differences
  - Used non threaded flavor of Globus
  - Smaller protocol buffer
  - Driver operates on vectors as opposed to buffers