S02: A Tutorial Introduction to High Performance Data Transport

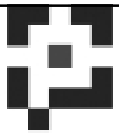
November 16, 2003
SC 2003
Phoenix, AZ

William E. Allcock
Argonne National Laboratory

Robert Grossman
University of Illinois at Chicago &
Open Data Partners

Steve Wallace
Indiana University

Table of Contents



pervasivetechnologylabs

AT INDIANA UNIVERSITY

Chapter 1
Review of Current Transport Protocols

Steve Wallace
Indiana University

Outline

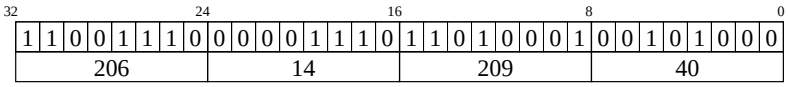
1. Review of the TCP/IP stack
2. Overview of TCP
3. TCP performance issues
4. Improvements to TCP
5. Performance of current TCPs
6. Congestion avoidance
7. Overview of UDP
8. Examples of UDP
9. SQL Slammer's use of UDP

Review of the TCP/IP Stack

Application
Presentation
Session
Transport
NETWORK
Data Link
Physical

- The Internet Protocol (IP) is at the *network layer* of the protocol stack
- IP provides *unique, global* addresses for hosts
- Hosts are located with their IP addresses

Format of an IP Address



206 . 14 . 209 . 40

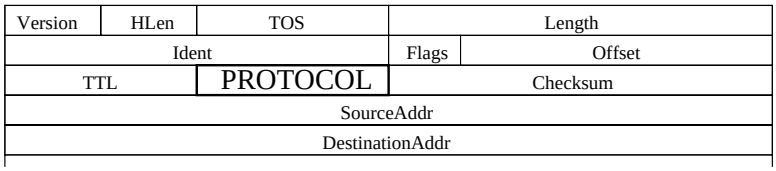
- An IP address is *32 bits long* and is usually written in *dotted decimal* notation.
- Internet routing is based on the leftmost bits of the destination IP address in a packet.

Transport Protocols

Application
Presentation
Session
TRANSPORT
Network
Data Link
Physical

- Actual conversations between hosts use a *transport* protocol on top of IP
- These can be reliable or unreliable, stream-oriented or message-oriented
- Most common are TCP and UDP

IP and Transport Protocols



- The *protocol field* in the IP header specifies which transport protocol is being used.
- This is an 8-bit field, so there are 256 possibilities.
- TCP is protocol 6, UDP is protocol 17.

Port Numbers

- Both TCP and UDP headers include a 16-bit *port number*.
- Network applications use "well-known" port numbers (e.g., HTTP is on port 80).
- A conversation is identified by the combination of *IP address*, *protocol*, and *port number* on both the source and destination hosts.

Common Port Assignments

Transport Protocols for Common Services

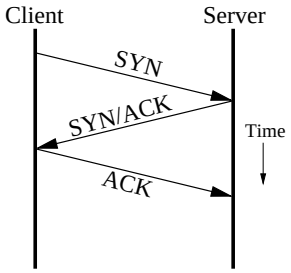
Service	Protocol	Port Number
FTP	TCP	21
SSH	TCP	22
SMTP	TCP	25
DNS	UDP or TCP	53
TFTP	UDP	69
HTTP	TCP	80

Overview of TCP

- TCP provides a *reliable*, *stream-oriented* connection between two hosts.
- What this means to applications:
  - *Reliable*: The transport protocol will take care of transmission and error detection itself.
  - *Stream-oriented*: The data is read in the order it was transmitted and has no "boundaries" other than those imposed by the application itself.

Three-Way Handshake

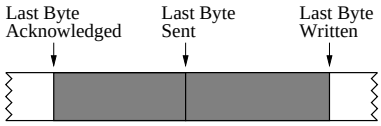
- TCP connections begin with a *three-way handshake*.
- This means each TCP connection takes three *round-trip times* (RTTs) to establish.
- This is the reason for things like HTTP "keep-alive".



Sliding Window Algorithm

- The core of TCP is the *sliding window algorithm*, which:
  - Guarantees reliable delivery of data.
  - Guarantees in-order delivery of data.
  - Provides mechanisms for flow control.
- This works *much* better than "send, wait for ACK, send, wait for ACK…" systems.
- Enhancements to TCP use the properties of this algorithm to try to "keep the pipe full".

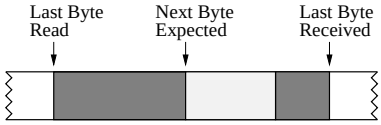
The Send Window



TCP Send Window

- For the *sending* part of a connection on a host, TCP maintains a buffer and three pointers:
  - The index of the *last byte acknowledged* by the remote host.
  - The index of the *last byte sent* out on the network.
  - The index of the *last byte written* to the buffer by the application.

The Receive Window

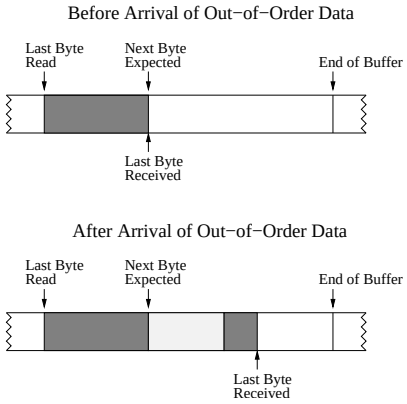


TCP Receive Window

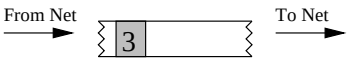
- For the *receiving* part of a connection on a host, TCP maintains another buffer and three pointers:
  - The index of the *last byte read* by the application.
  - The index of the *next byte expected* from the network by TCP.
  - The index of the *last byte received* from the network by TCP.

Out-of-Order Data

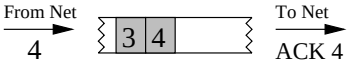
- When the receiver gets out-of-order data:
  - Anything not between the *next byte expected* and *end of buffer* is discarded.
  - Anything in between them is *kept*, but cannot yet be acknowledged.
  - We do send an ACK for the *existing* data.



Out-of-Order Packets Cause Duplicate ACKs

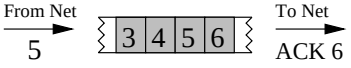


Connection has progressed to packet 3.

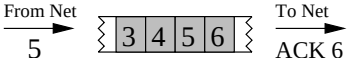


Packet 4 arrives; receiver ACKs packet 4.

Packet 6 arrives out of order; receiver keeps i and ACKs packet 4 again (fast retransmit).



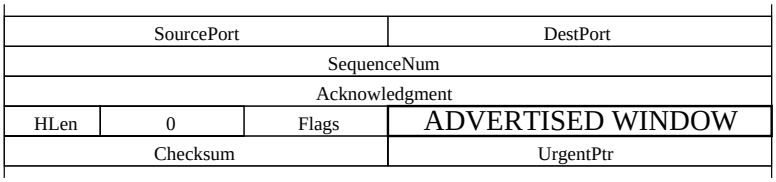
Packet 5 arrives out of order; receiver can now ACK packet 6.



Retransmission of packet 5 arrives; receiver ACKs packet 6 again.

Advertised Windows

- The hosts in a TCP connection inform each other of their current *receive* window size.
- This is the *advertised window* field in the TCP header; it is 16 bits wide.



Flow Control

- TCP implements *flow control* by maintaining invariants for the send and receive windows.
- Applications block on a *read* of an empty receive window, or a *write* to a full send window.
- The buffer size in the TCP stack does not change during the connection.

Window Size

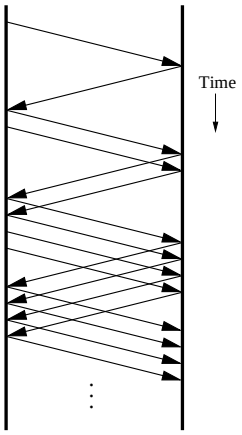
- The advertised window means, "You can send me this many bytes before I hear from you again."
- To keep the pipe full, we want this window to be equal to the *bandwidth-delay product* (BT*RTT)
- Examples:
  - 10 Mbps, 2 ms → 2.5 KB
  - 155 Mbps, 30 ms → 567.6 KB
  - 622 Mbps, 100 ms → 7.4 MB

Slow Start

- We can't have the full window open when we start a TCP connection.
  - We don't *know* the end-to-end bandwidth.
  - TCP/IP has no direct way of *telling* us the available bandwidth, which can change anyway.
- We don't want to start blasting 30 Mbps of data onto a congested link that can't handle the additional traffic.

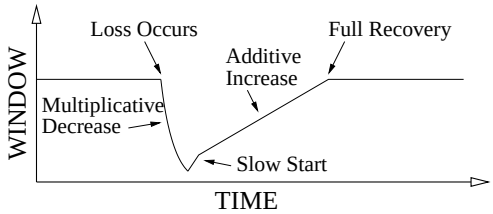
Illustration of Slow Start

- We start with a one-packet window.
- When the packet is ACKed, we double the window to two packets.
- Then to four packets, and so on, until we encounter loss or reach the buffer size.



Congestion Control

- All loss is attributed to network congestion.
- To avoid “unfriendly” behavior during congestion, we *multiplicatively decrease* the window size.
- We reenter slow start briefly and then *additively increase* the window size to recover.



…and much, much more

- TCP is a complex protocol.
  - There are 12 states in the state diagram.
  - The base RFC for TCP is 172 KB long.
- TCP has been the subject of research and performance tuning for over 20 years.
- There are very few independent implementation of TCP.

TCP Performance Issues

- For high-performance data transfer, TCP is *not* a “fire and forget” protocol.
- There are a number of performance issues with TCP, especially on high-latency network connections.
- Many of this issues can be resolved with proper end-system configuration.

"Congestion" Management

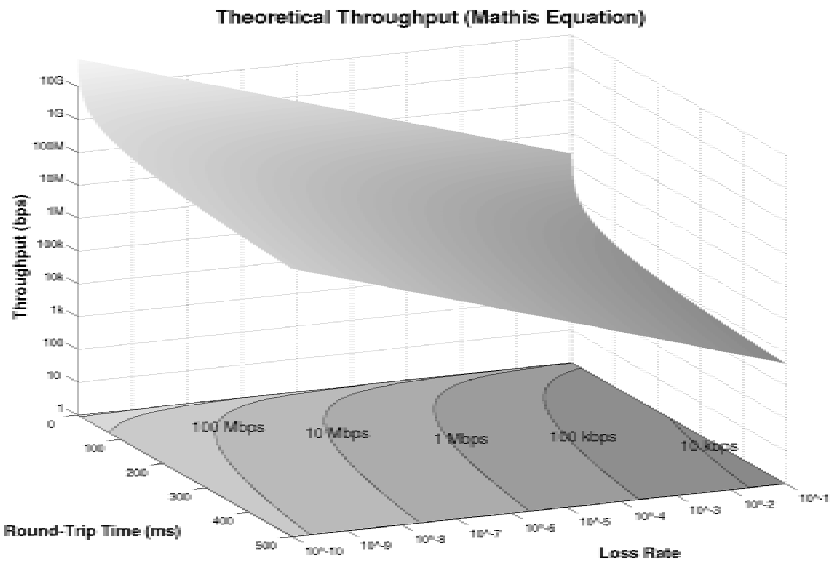
- We have seen that all loss in TCP is attributed to network congestion.
  - This has been true for most of the history of TCP.
- Now we have high-capacity, uncongested research networks that still have low-level packet loss.
  - This loss comes from cabling problems, connectors, etc.

The Mathis Equation

- This equation states that TCP *throughput* is limited by:
  - The *maximum segment size* (the amount of data in a single packet).
  - The *inverse* of the *round-trip time*.
  - The *inverse* of the *square root* of the *rate of packet loss*.

$$\text{Throughput} < \frac{\text{MSS}}{\text{RTT} \times \sqrt{\text{Loss}}}$$

The Mathis Equation Illustrated



What Does This Mean?

- We can improve throughput with *larger packets*, *lower latency*, or *lower packet loss*.
  - Larger packets require modifications to the network.
    - Over standard Ethernet, MSS is fixed at 1460 bytes.
  - Latency on international optical data networks is characterized by the speed of light.
  - Some low-level packet loss is nearly inevitable.
- All three are hard, but the first is easiest.

Requirements for 10 Gbps Flows

- To maintain this rate, we can lose only one packet every five hours.
  - This is *orders of magnitude* less than our very best wide-area data networks.
- Recovering from a single retransmission takes around five minutes.
- The target window size with 100 ms latency is around 120 MB.

Other Performance Issues

- The three-way handshake and slow start are problematic for large numbers of brief TCP sessions.
  - This is the primary characteristic of Web traffic.
- The default window size on many OSes is too small for good performance.
- Some TCP implementations are handicapped by having to copy TCP data several times.

Improvements to TCP

- There are quite a few suggested improvements to TCP.
- We will take a quick look at a few of them:
  - Large MTU
  - Large windows
  - Faster startup
  - SACK
  - ETEN

Large MTU

- This is not actually an improvement to TCP itself.
- Standard Ethernet frames are at most 1500 bytes long.
  - This has remained constant as Ethernet has gone from 10 Mbps all the way to 10 Gbps.
- A larger MTU allows for a larger MSS in the Mathis equation and thus (in theory), better throughput.

Large Windows

- Recall that the advertised window field in the TCP header is only 16 bits wide.
  - The TCP window is thus limited to 64 KB.
- RFC 1323 describes a TCP extension for supporting larger windows.
- We interpret the window size in the header as having been reduced by some factor.

Faster startup

- Modifying TCP slow start to have a slightly larger initial window improves throughput for brief connections.
  - In other words, start with a 2- or 3- packet windows instead of a 1-packet window.
- This can greatly improve performance for applications which use many short-lived connections.
  - In other words, the web and web services.

SACK

- SACK stands for *selective acknowledgment.*
- With large windows, there may be multiple packets lost within the window.
  - Standard TCP with a cumulative ACK can only inform the sender of one packet loss per RTT.
- We can do better by giving the sender a list of missing segments.
- This is defined in RFC 2018.

D-SACK

- This is an RFC 2018-compatible extension to SACK that uses SACK to report the reception of duplicate contiguous segments of data.
  - This allows the sender to learn the order in which packets are received.
  - This information can be used to detect out-of-order packet transmission, ACK loss, and packet replication.
- This was made official in RFC 2883.

ETEN

- ETEN stands for *Explicit Transport Error Notification*.
- Covers a variety of approaches:
  - Forward or backward error notification
  - Per-packet or cumulative error information
- Basic goal is to allow TCP to know what loss is due to congestion and what loss is due to the network itself.

Performance of Current TCPs

- We say "TCPs" because there are now a number of research TCP stacks:
  - TCP/Tahoe
  - TCP/Reno
  - TCP/NewReno
  - TCP/Vegas
  - And others…
- Each of these has different performance characteristics.

Common Properties

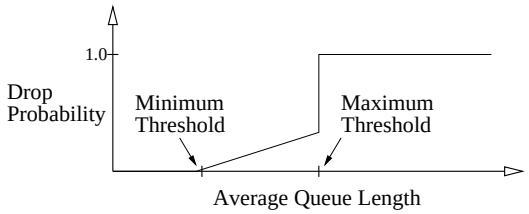
- Performance tracks the Mathis equation reasonably closely.
- Performance is characterized by the worst hop in the connection.
- Considerable end-system tuning is needed for best performance.
- Last-mile problems often dominate real-world performance.

Congestion Avoidance

- We can maintain higher aggregate TCP throughput if we are able to react to network congestion *before* it happens and causes us to lose many packets.
- Two main approaches:
  - Host-based congestion prediction (TCP-Vegas)
  - Router-based detection (RED)
- We will take a quick look at RED.

Random Early Detection

- Basic idea: Provide hosts with *implicit* notice of impending congestion by dropping a small number packets *before* the router queues are full.



Issues with RED

- RED may cause routers to favor "excess" UDP traffic over "excess" TCP traffic.
  - Many UDP-based applications lack any sort of congestion detection or control.
  - The TCP traffic gets punished for being more intelligent.
- It may be advisable to have different RED thresholds for different types of traffic.

Overview of UDP

- UDP provides a *best-effort*, *datagram-oriented* connection between two hosts.
- What this means to applications:
  - *Best-effort*: Messages may be lost between the source and destination, and the transport protocol does not retransmit any data.
  - *Datagram-oriented*: There is no order associated with messages, and they may arrive in any sequence.

UDP is Simple

- UDP adds only *port numbers*, *length*, and a *checksum* to the IP header.
- Anything more than that is completely up to the application in question.
- Needless to say, UDP *is* easy to implement. (That's why TFTP exists, for example.)

SourcePort	DestPort
Length	Checksum

Examples of UDP

- UDP is traditionally used for applications with some special needs:
  - Multicast
    - connections don't make sense
  - Streaming audio and video
    - reliable delivery isn't necessary
  - Brief, low-latency connections
    - three-way handshake is time-consuming
  - Protocol code that fits in a small amount of memory
    - TCP is complex and difficult to reimplement

UDP-based Applications

- Multicast uses UDP because it has to.
- Streaming audio and video use UDP because some data loss is more acceptable than retransmission delays.
- DNS uses UDP for most queries in order to reduce response time and avoid creating state for client connections.
- TFTP uses UDP because it is easy to implement in limited space as a bootstrap loader.

SQL Slammer's Use of UDP

- The infamous "SQL Slammer" worm used UDP as a transport protocol – why?
  - The code for the worm fit entirely within a single UDP datagram.
  - This allowed one-way communication; no return traffic of any kind was necessary.
  - Using TCP would have required a transmission path back to the infecting host.
  - The infecting host could thus be anonymous.

Chapter 2
Striped TCP

Bill Allcock
Argonne National Laboratory

Definitions

- Logical Transfer
  - The transfer of interest to the initiator, i.e. move file foo from server A to server B.
- Network Endpoint
  - In general something that has an IP address. A Network Interface Card (NIC).
- Parallel Transfer
  - Use of multiple TCP streams between a given pair of network endpoints during a logical transfer
- Striped Transfer
  - Use of multiple pairs of network endpoints during a logical transfer.

What's wrong with TCP?

- You probably wouldn't be here if you didn't know that.
- TCP was designed for Telnet / Web like applications.
- It was designed when T1 was a fast network, big memory was 2MB, not 2 GB, and a big file transfer was 100MB, not 100GB or even Terabytes.

AIMD and BWDP

- The primary problems are:
  - Additive Increase Multiplicative Decrease (AIMD) congestion control algorithm of TCP
  - Requirement of having a buffer equal to the Bandwidth Delay Product (BWDP)
  - The interaction between those two.
  - We use parallel and striped transfers to work around these problems.

AIMD

- To the first order this algorithm:
  - Exponentially increases the congestion window (CWND) until it gets a congestion event
  - Cuts the CWND in half
  - Linearly increases the CWND until it reaches a congestion event.
  - This assumes that congestion is the limiting factor
  - Note that CWND size is equivalent to Max BW

BWDP

- Use a tank as an analogy
- I can keep putting water in until it is full.
- Then, I can only put in one gallon for each gallon removed.
- You can calculate the volume of the tank by taking the cross sectional area times the height
- Think of the BW as the area and the RTT as the length of the network pipe.

Recovery Time

$$\text{Recovery Time} = \frac{\text{Bytes to Recover}}{\text{Rate of Recovery}}$$

$$\text{Bytes to Recover} \propto \frac{1}{2}\,\text{BW} * \text{RTT (BWDP)}$$

$$\text{Rate of Recovery} \propto \frac{\text{MTU}}{\text{RTT}}$$

$$\text{Recovery Time} \propto \frac{\frac{1}{2}\,\text{BW} * \text{RTT}}{\frac{\text{MTU}}{\text{RTT}}} \Rightarrow \frac{\text{RTT}^2 * \text{BW}}{\text{MTU}}$$

Recovery Time for a Single Congestion Event

- T1 (1.544 Mbs) with 50ms RTT ≅ 10 KB
  - Recovery Time (1500 MTU): 0.16 Sec
- GigE with 50ms RTT ≅ 6250 KB
  - Recovery Time (1500 MTU): 104 Seconds
- GigE to Amsterdam (100ms) ≅ 1250 KB
  - Recovery Time (1500 MTU): 416 Seconds
- GigE to CERN (160ms) ≅ 2000 KB
  - Recovery Time (1500 MTU): 1066 Sec (17.8 min)

How does Parallel TCP Help?

- We are basically cheating…. I mean we are taking advantage of loopholes in the system
- Reduces the severity of a congestion event
- Buffers are divided across streams so faster recovery
- Probably get more than your fair share in the router

Reduced Severity from Congestion Events

- Don't put all your eggs in one basket
- Normal TCP your BW Reduction is 50%
  - 1000 Mbs * 50% = 500 Mbs Reduction
- In Parallel TCP BW Reduction is:
  - Total BW / N Streams * 50%
  - 1000 / 4 * 50% = 125 Mbs Reduction
- Note we are assuming only one stream receives a congestion event

Faster Recovery from Congestion Events

- Optimum TCP Buffer Size is now BWDP / N where N is number of Streams
- Since Buffers are reduced in size by a factor of 1/N so is the recovery time.
- This can also help work around host limitations. If the maximum buffer size is too small for max bandwidth, you can get multiple smaller buffers.

More than your Fair Share

- This part is inferred, but we have no data with which to back it up.
- Routers apply fair sharing algorithms to the streams being processed.
- Since your logical transfer now has N streams, it is getting N times the service it otherwise normally would.
- I am told there are routers that can detect parallel streams and will maintain your fair share, though I have not run into one yet.

What about Striping?

- Typically used in a cluster with a shared file system, but it can be a multi-homed host
- All the advantages of Parallel TCP
- Also get parallelism of CPUs, Disk subsystems, buses, NICs, etc..
- You can, in certain circumstances, also get parallelism of network paths
- This is a much more complicated implementation and beyond the scope of what we are primarily discussing here.

Nothing comes for free…

- As noted earlier, we are cheating.
- Congestion Control is there for a reason
- Buffer limitations may or may not be there for a reason
- Other Netizens may ostracize you.

Congestion Control

- Congestion Control is in place for a reason.
- If every TCP application started using parallel TCP, overall performance would decrease and there would be the risk of congestive network collapse.
- Note that in the face of no congestion parallel streams does not help
- In the face of heavy congestion, it can perform worse.

Buffer Limitations

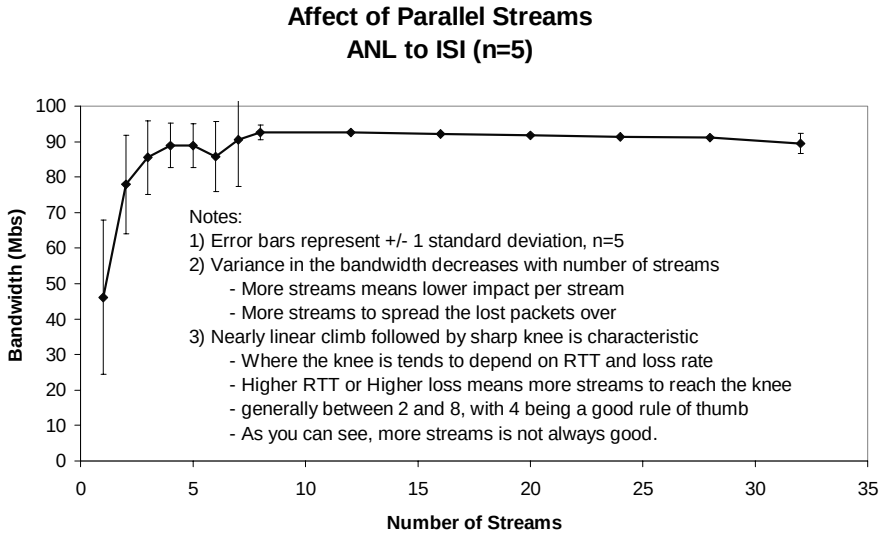
- More often than not, the system limitations are there because that is way it came out of the box.
- It requires root privilege to change them.
- However, sometimes, they are there because of real resource limitations of the host and you risk crashing the host by over-extending its resources.

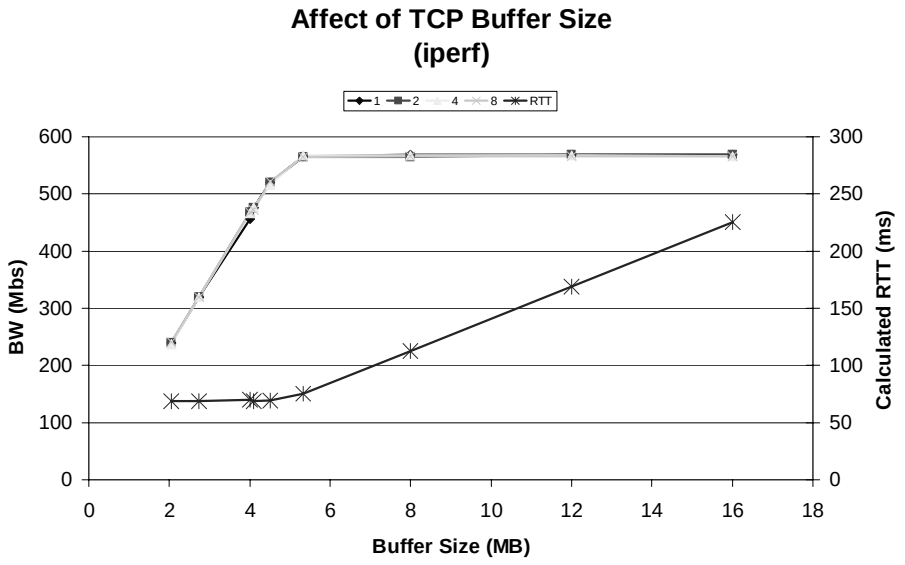
Cheat enough, but not too much

- If your use of parallel TCP causes too many problems you could find yourself in trouble.
  - Admins get cranky when you crash their machines
  - Other users get cranky if you are hurting overall network performance.
- Be a good Netizen

When should you use Parallel TCP?

- Engineered, private, semi private, or very over provisioned networks are good places to use parallel TCP.
- Bulk data transport. It makes no sense at all to use parallel TCP for most interactive apps.
- QOS: If you are guaranteed the bandwidth, use it
- Community Agreement: You are given permission to hog the network.
- Lambda Switched Networks: You have your own circuit, go nuts.





Chapter 3
Reliable UDP Protocols

Robert Grossman
University of Illinois at Chicago &
Open Data Partners

Yunhong Gu
University of Illinois at Chicago

Chapter 3 Table of Contents

3.1 Introduction

The Problem

- Johnson's Law is steeper than Moore Law
  - i.e., cost of disk declining faster than cost of cpu's
- All disk is quickly filled
- TCP simply not effective for many data intensive applications
- We need something today – cannot wait for new network infrastructure that some new protocols require
- This is not just about "big data" but also about working interactively with remote and distributed data

Trans-Atlantic TCP Performance Poor

- Consider 1 Gbps Chicago to Amsterdam link
  - 110ms RTT
- TCP with default settings
  - 5Mbps using default 64KB buffer
- TCP with BDP window tuning:
  - 30Mbps using turned 12MB buffer (=1Gbps*110ms)
- Parallel TCP
  - 300Mbps
  - 64 TCP concurrent flows, each using 12MB buffer.

Some Reasons for TCP's Poor Performance on High BDP Links

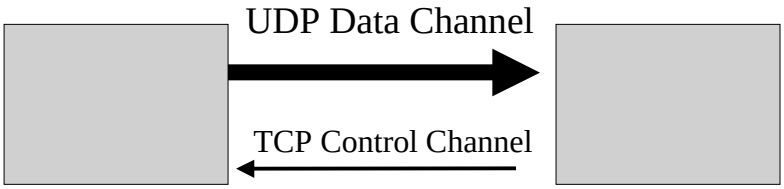
- Discover/recover slow on high BDP links
  - Increase 1 packet per RTT
  - Recovering on 10 Gb/s link can take minutes from single re-transmitted packet
- Drastic multiplicative decrease in sending rate
- Fairness bias against high BDP flows
- Link error more common in high BDP links which limits effective throughput via Mathis Equation

Desirable Properties for High Performance Data Transport Protocol

- Efficiency
  - High utilization of the abundant bandwidth either with single or multiplexed connections
- Fairness (intra-protocol fairness)
- Fair to both low and high BDP flows
- Friendly to TCP flows
- Easy to deploy
  - ideally requires no change to network infrastructure

3.2 Basic Ideas for UDP Data / TCP Control Protocols

Separate Data and Control Channels



- Separate data and control channels
- Use UDP channel for data
- Use TCP channel for control - reliability, congestion control, & flow control
- Rely on many years spent building good TCP and UDP implementations

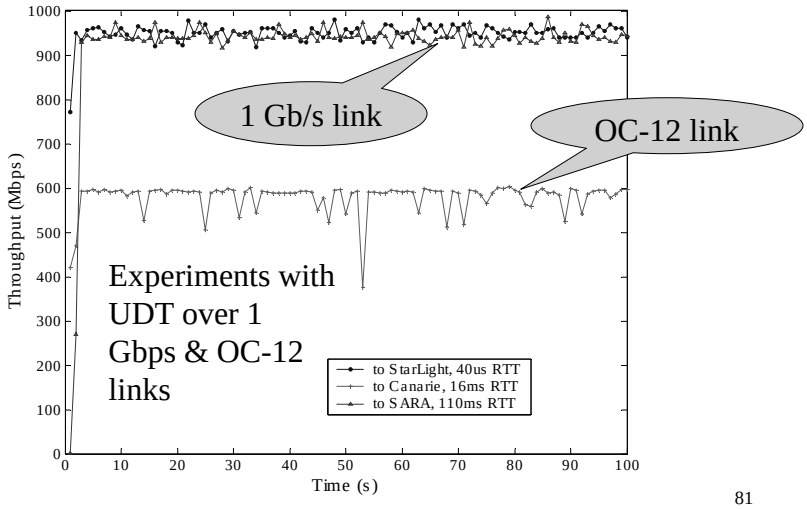
Advantages

- Fast - effective use of available bandwidth
  - no striping required to fill 1 Gb/s links
- Easy to deploy since application level:
  - no changes in routers
  - no changes in operating systems
- Easy to use since no manual tuning of windows required
  - remember filling the BDP pipe with TCP requires window tuning

Challenges

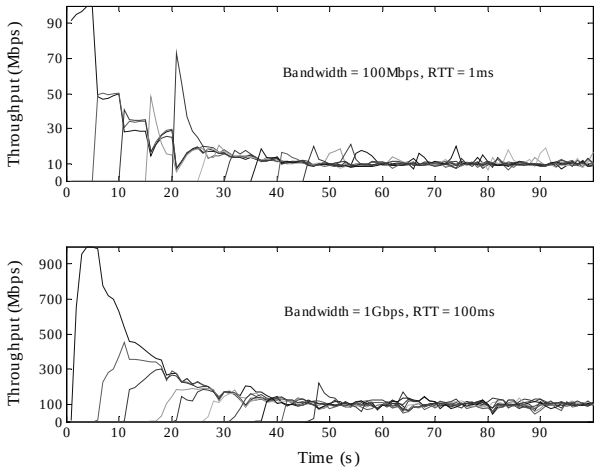
- Intra-protocol fairness must be established for new protocols
- TCP-friendliness must be established for new protocols
  - UDP has caused problems in the past
- We know that TCP is not fair to flows with high BDP
- Conservative community may not accept new protocols
- Must develop stable implementations

UDP Data / TCP Control Protocols Can be Fast



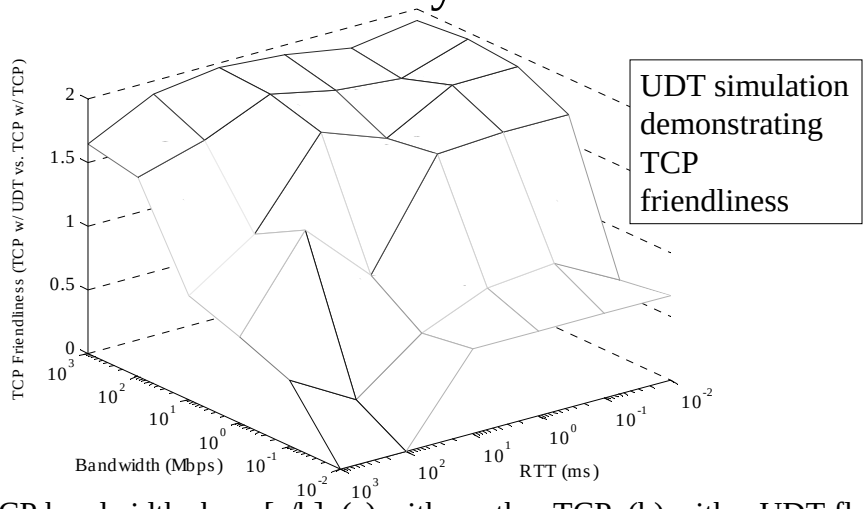
UDP Data / TCP Control Protocols can Be Fair

UDT simulation results demonstrating intra-protocol fairness



10 concurrent UDT flows share 100Mbps and 1Gbps links

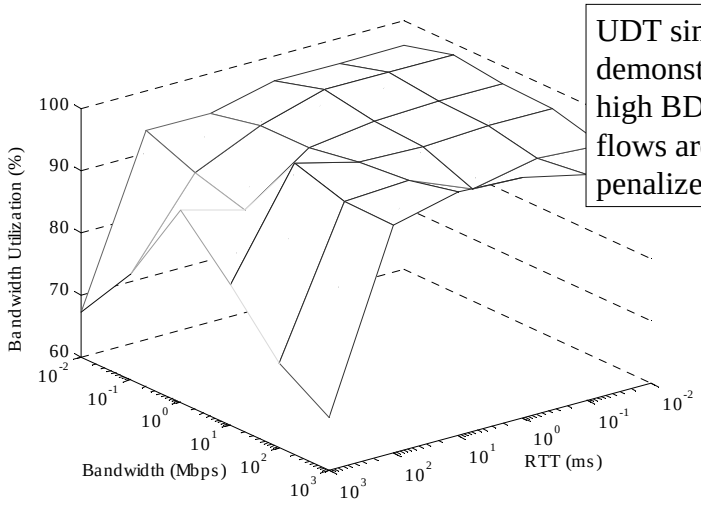
UDP Data / TCP Control Protocols can Be Friendly to TCP Flows



UDT simulation demonstrating TCP friendliness

TCP bandwidth share [a/b]: (a) with another TCP, (b) with a UDT flow

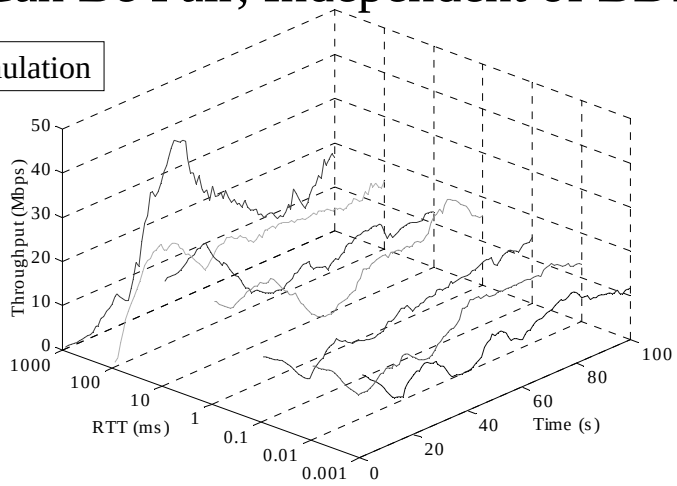
UDP Data / TCP Control Protocols Can Be Fair, Independent of BDP



UDT simulation demonstrating high BDPs flows are not penalized.

UDP Data / TCP Control Protocols Can Be Fair, Independent of BDP

UDT simulation



7 concurrent UDT flows with different RTTs sharing 100Mbps link

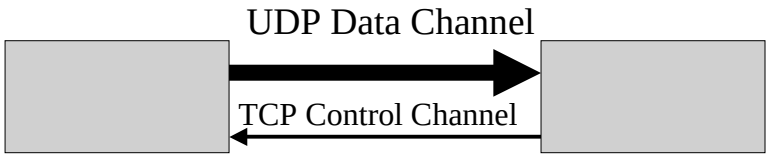
3.3 Implementations of UDP Data / TCP Control Protocols

Multiple Implementations Available

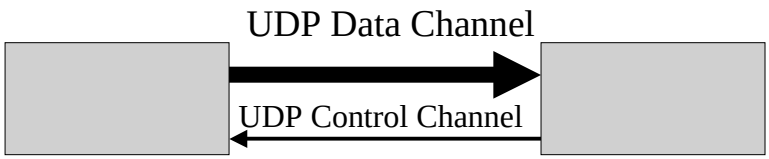
	Rate Control	iGrid 02 Performance	Open Source	Research Group
SABUL	AIMD	940 Mb/s	Yes	UIC - LAC
Tsunami	Constant Rate	769 Mb/s	Yes	IU
RPUDP	Constant Rate	680 Mb/s	Yes	UIC - EVL
FOBS	Not stated	NA	Yes	ANL

3.4 Case Study
SABUL / UDT

SABUL & UDT



SABUL: Simple Available Bandwidth Utilization Library



- UDT: UDP based Data Transfer

SABUL/UDT Characteristics

- Reliable, UDP Data / UDP or TCP Control Protocol
- Can be deployed at application level
- Fair and Friendly:
  - intra-protocol fairness
  - TCP friendliness
  - and RTT independence.
- Implementation: Open source C++ library available via Source Forge

Why Does UDT use a UDP Control Channel?

- NO TCP connection required
- More flexible and can be implemented on any connectionless packet switch network.
- Easier to develop friendly protocols that effectively use bandwidth in all UDP environments (not mixed UDP and TCP flows)

SABUL Protocol Overview

- Uses both Rate Control (RC) and (window based) Flow Control (FC)
  - Constant *RC interval* to remove RTT bias
  - Employs *bandwidth estimation*
- Selective acknowledgement (ACK)
  - Reduces control traffic & results in faster recovery
- Uses *packet delay* as well as packet loss to indicate congestion
- Slow start
  - controlled by FC

SABUL/UDT Acknowledgements

- Selective acknowledgement (ACK)
  - Generated at every constant interval (same as SYN) to send back largest continuously received sequence number of data packets.
  - The sender sends back an ACK2 to the receiver for each ACK (sub-sequence/ack).
- Explicit negative acknowledgement (NAK)
  - Generated as soon as loss is detected.
  - Loss information may be resent if receiver has not received the retransmission after some increasing interval.
  - Loss information is compressed in NAK.

SABUL/UDT Congestion Control

- Using both Rate Control and (window based) Flow Control
  - Rate control (RC): triggered at every *constant interval* at sender side.
  - Constant RC interval, or synchronization interval (SYN): 0.01 seconds.
  - Flow control (FC): triggered at each received ACK at sender side.
- UDT is packet based protocol (NOT byte based).
  - Packet has the same size as MTU.

Congestion Control (cont.)

- Packet *delay* is detected and used as one of the indications of congestion
  - The delay increasing trend is detected by variation of RTT at receiver side and sent back with a special warning message. No more than 1 warning message is generated in 2 RTTs.
  - Reduce loss.
  - Reduce persistent queue size.

SABUL/UDT Rate Control

- Control the *inter-packet* time.
- AIMD:
  - Increase parameter is related to *link capacity* and current sending rate
  - Decrease factor is 1/9, but does not decrease for *all* loss events
- Link capacity is *probed by packet pair*, which is sampled UDT data packets.

SABUL/UDT Flow Control

- Triggered when receiving an ACK.
- Limit the number of unacknowledged packets (W).
- W = W*0.875 + *AS**(RTT+SYN)*0.125
- *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.

SABUL/UDT Slow Start

- Flow window starts at 2 and increases to the number of acknowledged packets, until the sender receives an NAK or a delay warning message, when slow start ends.
- Slow start only occurs at the beginning of a UDT session.
- Inter-packet time is 0 during slow start phase and set to the packet arrival interval at the end of the phase.

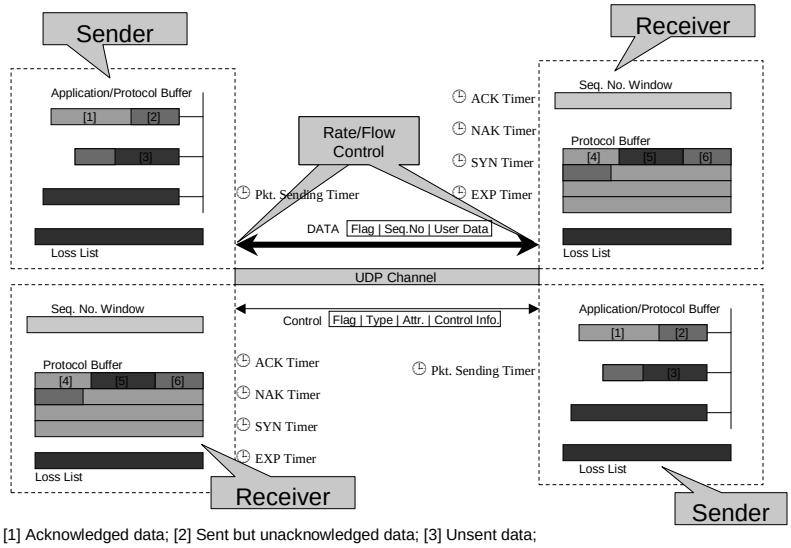
SABUL/UDT Performance

- Convergence: Discovers bandwidth and reacts to congestion quickly
- Lightweight: low packet header and computational overhead
- Avoids congestion collapse
- Selective acknowledgement means less control traffic
- Robust to network changes

SABUL/UDT Fairness

- Intra-Protocol Fairness: All SABUL/UDT flows should share similar bandwidth.
- TCP-Friendly: Shares bandwidth with TCP in low BDP networks where TCP works fine;
- Leaves TCP enough space to increase in high BDP links where TCP cannot fully utilize the bandwidth.
- Independent of network delay.

UDT Arch.



[1] Acknowledged data; [2] Sent but unacknowledged data; [3] Unsent data;
[4] Unwritten buffer; [5] Received and acknowledged data; [6] Received but unacknowledged data.

3.5 SABUL/UDT Experimental Results

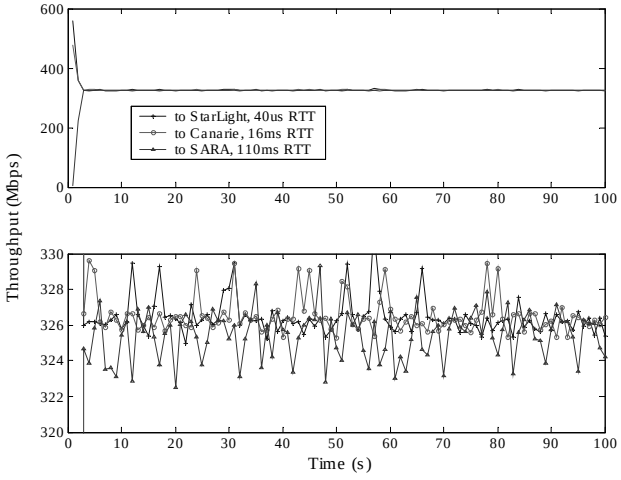
Striped SABUL at iGrid 02

TCP Stream	GridFTP	SABUL Stream 1	SABUL Stream 2	SABUL Stream 3	Striped SABUL Stream
4.36 Mb/s	324 MB/s	902.8 Mb/s	902.9 Mb/s	907.1 Mb/s	2712.8 Mb/s

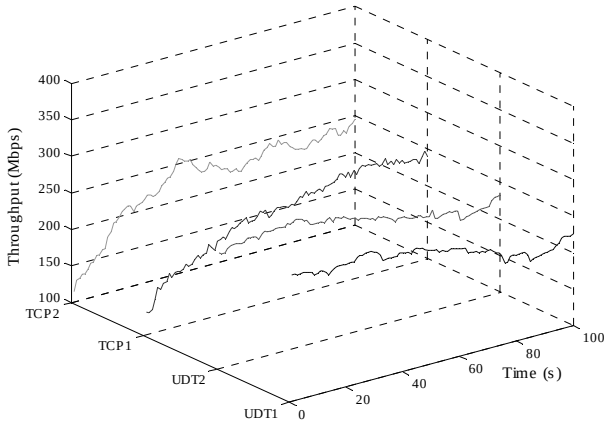
- Three node cluster in Chicago connected to three node cluster in Amsterdam connected with 10 Gb/s link and 110 ms RTT

UDT Experimental Results Demonstrating Intra-protocol Fairness

3 concurrent UDT flows with different RTTs sharing 1Gbps link

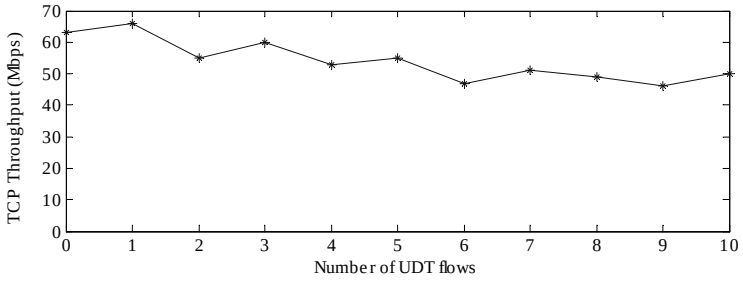


UDT Experimental Results Demonstrating TCP Friendliness



2 TCP and 2 UDT flows coexist on 1Gbps link

UDT is TCP Friendly with 500 concurrent TCP flows



500 TCP flows with different number of UDT backgrounds

3.6 Summary

Summary

- UDP Data / TCP Control protocols
  - are effective even in high BDP links
  - are fair to other UDP/TCP flows
  - are friendly to TCP flows
- No change to existing routers or OS required
- No manual tuning required

Summary (cont.)

- There are multiple implementations available
- There are preliminary plans to work on common code releases
- Open source – No IP restrictions
- Community's verdict still out
- Meanwhile, use something that works

For More Information

- UIC's SABUL/UDT
  - Web site: www.lac.uic.edu
  - UDT source code on Source Forge
  - www.sourceforge.net (Project Name: DataSpace)
  - Linux, BSD and Unix implementations
  - NS-2 simulation code available
- IU's Tsunami
  - Web site: www.anml.iu.edu/anmlresearch.html
  - open source

References (1)

1. D. Katabi, M. Hardley, and C. Rohrs: Internet Congestion Control for Future High Bandwidth-Delay Product Environments, ACM SIGCOMM 2002, Pittsburg, PA.
2. T. V. Lakshman and U. Madhow: The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. IEEE/ACM Trans. on Networking, vol. 5 no 3, July 1997, pp. 336-350.
3. Foster, C. Kesselman, S. Tuecke: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001.
4. W. Feng and P. Tinnakornsrisuphap: The Failure of TCP in High-Performance Computational Grids, Proc. of SuperComputing 2002.
5. H. Sivakumar, S. Bailey, R. L. Grossman: PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks. Proc. of SuperComputing 2000.
6. B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke: Data Management and Transfer in High Performance Computational Grid Environments. Parallel Computing Journal, Vol. 28 (5), May 2002.
7. Hacker, T., Athey, B., and Noble, B.: The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS) 2001.

References (2)

8. D. Clark, M. Lambert, and L. Zhang: NETBLT: A high throughput transport protocol, SIGCOMM '87, (Stowe, VT).
9. D. Cheriton: VMTP: A transport protocol for the next generation of communication systems, SIGCOMM '87, (Stowe, VT).
10. Strayer, T., Dempsey, B., and Weaver A.: XTP – the Xpress Transfer Protocol. Addison-Wesley Publishing Company, 1992.
11. H. Sivakumar, R. L. Grossman, M. Mazzucco, Y. Pan, and Q. Zhang: Simple Available Bandwidth Utilization Library for High-Speed Wide Area Networks, Journal of Supercomputing, to appear.
12. T. Dunigan, M. Mathis and B. Tierney: A TCP Tuning Daemon, Proc. of IEEE SuperComputing 2002.
13. UDT source code. http://sourceforge.net/projects/dataspace/.
14. HighSpeed TCP. http://www.icir.org/floyd/hstcp.html.
15. Tsunami. http://www.anml.iu.edu/anmlresearch.html.
16. E. He, Leigh, J., Yu, O., and DeFanti T. A.: Reliable Blast UDP: Predictable High Performance Bulk Data Transfer. IEEE Cluster Computing 2002, Chicago, IL, Sep. 2002.
17. P. Dickens: FOBS: A Lightweight Communication Protocol for Grid Computing. To Appear on Europar 2003.
18. FAST TCP. http://netlab.caltech.edu/fast.

Chapter 4
Beyond TCP

Steve Wallace
Indiana University

Outline

1. Conditions in R&E networks
2. Parallel / Striped TCP
3. Hybrid protocols
4. Special-case applications
5. Huge MTUs
6. Operating system support
7. What's in the future?

Conditions in R&E Networks

- Different conditions prevail in research and education networks than in the commodity Internet.
  - The networks are often running far below peak capacity.
    - Abilene's busiest link is often around 700 Mbps of data—on an OC-192 circuit.
  - Data transfer is seldom billed "by the byte".
  - The challenge is trying to use more of the pipe, not trying to force application to fit in it.

What This Means for TCP

- Recall from Chapter 1 that TCP attributes packet loss to congestion.
- Uncongested R&E networks have unavoidable low-level packet loss due to equipment and cabling.
  - This causes TCP congestion control to kick in.
  - Throughput collapses.
- The ultra-low levels of packet loss necessary to support 10 Gbps TCP transfers cannot be achieved with current technology.

Recall the Mathis Equation

$$\text{Throughput} < \frac{\text{MSS}}{\text{RTT} \times \sqrt{\text{Loss}}}$$

- For any single TCP stream, we can improve performance by:
  - Increasing the *maximum segment size*.
  - Decreasing the *round-trip time*.
  - Decreasing the *rate of packet loss*.
- …Or we can try something other than a single TCP stream.

Parallel / Striped TCP

- Instead of a single TCP stream, run a number of TCP streams in parallel.
- Although each stream is susceptible to the problems we've explored, at any point in time, most of the streams should be doing fine.
- For example, bbFTP and some popular peer-to-peer file sharing applications take this approach.

Advantages of Parallel TCP

- Easy to implement
  - No protocol or OS changes are necessary; the implementation can lie entirely in user space.
- Uses a well-understood transport protocol.
  - The behavior of one of the individual TCP streams will be fairly predicable.
- Can be used by degree.
  - Easy to go from 2 streams to 3, or 4, or …

Disadvantages of Parallel TCP

- Concerns about fairness.
  - 1,000 concurrent TCP streams generally have 1,000 times as many "votes" as a single stream.
- Greater overhead on end systems.
  - Large TCP windows for each connection and possibly many context switches and disk seeks.
- Every session may collapse simultaneously.
  - 1500 bytes is 1 μs at 10 Gbps rates.
  - An event that causes loss can easily last enough to affect multiple streams.

Hybrid Protocols

- Recall that UDP is much simpler than TCP.
  - No flow control.
  - No automatic retransmissions.
  - No congestion control.
- Except for the addition of port numbers and a checksum, UDP is just raw IP packets.
- UDP is not sensitive to the restrictions of the Mathis equation, just the IP stack itself.

Why Not Pure UDP?

- If UDP doesn't share these constraints, why not base more data transport applications on UDP?
  - Transmitting data reliably with IP requires administrative overhead: that's TCP.
  - Transmitting data reliably with UDP thus requires administrative overhead as well.
- We don't want to throw away TCP and its 20 years of experience and tuning.

UDP Data / TCP Control

- Solution: Combine UDP and TCP into a single transport protocol.
  - Use UDP for the (high-bandwidth) *application data.*
  - Use TCP for the (low-bandwidth) *control data*.
- UDP carries the payload and TCP carries error feedback, acknowledgements, retransmission requests, etc.

Hybrid Examples

- SABUL (Grossman, et. al.)
- Tsunami (Wallace, et. al.)

Both of these use the model of a high-speed UDP flow managed by information exchanged on a low-speed TCP flow.

Special-Case Applications

- UDP and TCP offer two different transport models for applications:
  - Best-effort and message-oriented
  - Reliable and stream-oriented
- What about other possibilities?
  - Best-effort and stream-oriented
  - Reliable and message-oriented
- Are these categories of meaning or value?

Best-Effort and Stream-Oriented

- This is easy to simulate with UDP and an extended packet header.
- Think of multicast applications, streaming audio and video data, and so forth…

Reliable and Message-Oriented

- Consider the properties of a file transfer from disk to disk:
  - The size of the file is known in advance.
  - The file is easily divided into blocks.
  - Each end host has random access to any part of the file.
- So out-of-order delivery is irrelevant if the blocks are tagged with their sequence number – we don't *need* a stream!
- Can be simulated with a hybrid protocol.

Huge MTUs

- As Ethernet speed has improved, the maximal packet size has not increased.
- The chart below shows the length of the MTU in transmission time and what the MTU *would* be if it had scaled with speed.

Ethernet Speed	1500−byte Transmission Time	MTU with Scaling
10 Mbps	1.2 ms	1.5 KB
100 Mbps	0.12 ms	15 KB
1 Gbps	12 us	150 KB
10 Gbps	1.2 us	1.5 MB

Advantages of a Larger MTU

- We get higher maximum throughput, according to the Mathis equation.
- We devote a smaller proportion of our bandwidth to packet and frame headers.
- We use less CPU time in processing the same amount of application data.
- We reduce the number of interrupts generated by incoming packets.

Disadvantages of a Larger MTU

- Requires support at every hop between the end hosts.
- Larger packets require more (expensive) buffer memory in routers.
  - If MTU *had* scaled, a 10-packet queue on a 10-Gbps interface is 15 MB!
- Jitter increases as packets are longer in duration.
- If errors are randomly distributed point events, each packet is more likely to hit one in proportion to its size.

Support for Large MTUs

- Ethernet jumbo frames: 9,000 bytes
- IPv4 packet size limit: 65,536 bytes
- IPv6 includes, as part of the standard, support for "jumbograms"
  - Jumbograms can be up to 4 GB in size.

Operating System Support

- Most modern Unix/Unix-like systems support large windows and SACK.
- Windows 95 (without Winsock 2) and Windows NT 4.0 *do not* support large windows and SACK.
- Windows 98, 2000, and XP include SACK support—and large windows, *if* you edit the registry.

What's in the Future?

- We can only speculate.
- Nevertheless, there are new ideas and trends that it would be a good idea to keep an eye on...

Storage-Conscious Transport Protocols

- The most difficult part of tuning a high-performance, high-volume transport protocol is the interaction between the network and disk subsystems in the OS.
- Look for transport protocols that are conscious of this interplay and optimized for disk-to-disk data transfers.

More Self-Tuning

- Today, getting good transport protocol performance depends on tuning the end systems.
- This task can demand considerable expertise in what may be an unrelated field.
- Expect to see a trend toward more intelligent, “self-tuning” network stacks.

Divergence of Transport Protocols

- The R&E networking world and the commodity networking world are becoming increasingly different.
- Moving 10,000 TB of high-energy physics data is not a common application of the commodity Internet.
- Expect to see transport protocols appropriate only for use on R&E-type networks.

Connection Retermination and Hop Elimination

- Some applications can get better throughput by terminating a TCP connection at a "network coupler" at some point within the network.
- We can't propagate signals *faster* than *c*, but we can get *closer* by eliminating hops.
  - This is a central idea of "lambda networking".

The Possible End of AIMD

- AIMD is the TCP-like behavior of *additive increase*, *multiplicative decrease* of window size in response to congestion.
- We have seen that the "AI" part can lead to very long recovery times.
- Expect to see transport protocols that use MIMD (or some logarithmic intermediary).

Composable Protocol Fragments

- Future transport protocols may not be a monolithic entity like TCP.
- Imagine constructing a transport protocol appropriate to your particular application by composing “protocol fragments” that implement different features.

Lambda Networking

- Meaning of the term in the context of R&E Networks
- Current implementations
- Coordination occurring internationally
- Currently developing more precise definitions

Chapter 5
Protocols for Moving Attribute Based Data

Robert Grossman
University of Illinois at Chicago
Open Data Partners

Yunhong Gu
University of Illinois at Chicago

Chapter 5 Table of Contents

5.1 Introduction

What is the Problem?

- Data is different than bits
- Data has attributes, data has metadata, data has keys, data can be merged, etc.

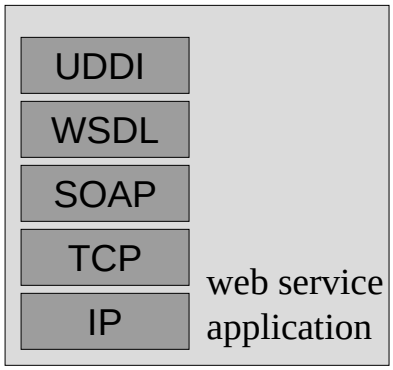
- By using specialized protocols for accessing & integrating data, data intensive applications can be built more easily
- With these protocols, distributed data mining applications are also easier to build

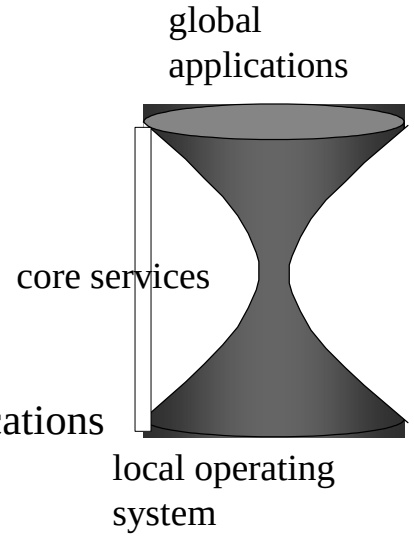
Central Role of Web Services

Data Grids	K. Grids	Data Webs	Semantic Web
Web Services	High Perf. WS	OGSA/DAI	

- Web services have emerged as the underlying infrastructure for a number of different distributed middleware platforms.
- Natural to develop high performance web services for data and to integrate web services and grid services

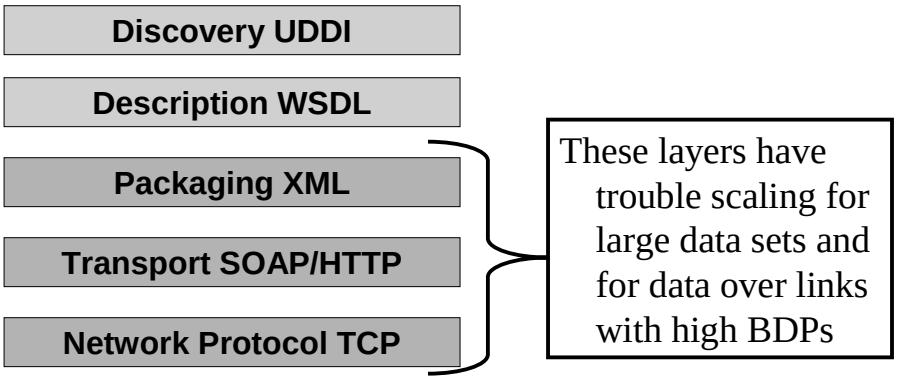
Basic Idea - Layers





- diverse global applications
- few core services

Observe Current Web Services do not Scale to Large Data Sets



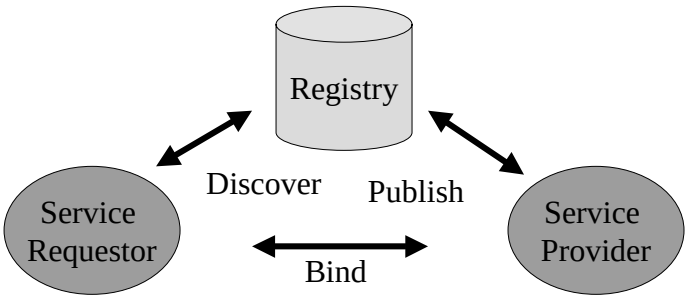
5.2 Web Services & SOAP/XML

Web Services Definition

"A Web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML based messages exchanged via internet-based protocols."

- www.w3.org

Service Based Architecture

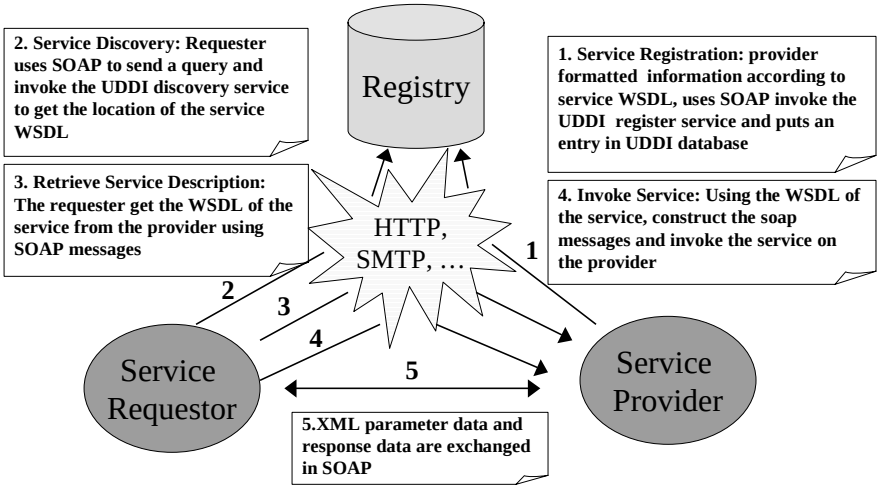


- Platform independent software component published via a directory or registry by a service provider
- Distributed computing paradigm that differs from DCE, CORBA, & Java RMI by exploiting internet protocols & XML

Web Service Elements

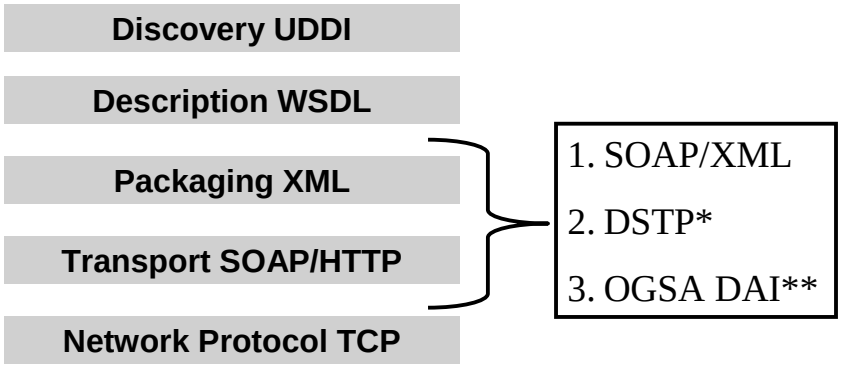
- UDDI: Discover and locate the web services
  - Universal Description, Discovery and Integration:
- WSDL - Describe the web services
  - Web Services Description Language
- XML - Package data
- SOAP - invoke the web services
- TCP - transport the query and request
- IP – relies on IP's unique, global addresses

Web Service Conversation



Source: Based in part on material from Isabel Cruz

Web Services Provide a Framework for Variety of Data Protocols



* DSTP is compliant with web services and OGSA

** DAI is data access and integration

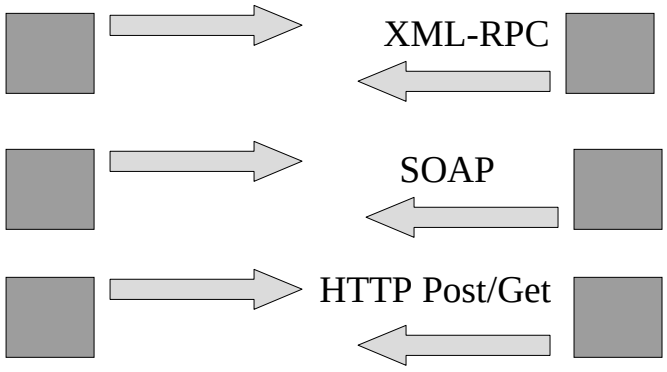
Web Services: A Better RPC Architecture

- Compared with other distributed computing platforms (CORBA, DCOM, and Java RMI) web services are:
  - Loosely coupled & can be invoked at run time
  - Communication endpoints can be URLs
  - Text based SOAP is wire protocol
  - XML used for payload
- Many consider to be easier to deploy and more flexible.

SOAP

- XML messaging provides an application and platform independent means of sharing data
- SOAP is a good mechanism for sending XML messages
- Focus to date is on using SOAP for sending XML-RPCs over HTTP
- SOAP messages consists of
  - SOAP envelope
  - SOAP header
  - SOAP body

XML Messaging



- XML good for metadata
- XML good for small data

Example: Google SOAP XML Query

```
<SOAP-ENV:Envelope
        xmlns:SOAPENV="http://schemas.xmlsoap.org/... >
   <SOAP-ENV:Body>
        <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"  … ">
        <key xsi:type="xsd:string">XXXXXXXXXXXX</key>
        <q xsi:type="xsd:string">  data  </q>
        <start xsi:type="xsd:int">0</start>
        <maxResults xsi:type="xsd:int">10</maxResults>
        <filter xsi:type="xsd:boolean">true</filter>
        …
        </ns1:doGoogleSearch>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Example: Google SOAP XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:doGoogleSearchResponse xmlns:ns1="urn:GoogleSearch" SOAP-
    ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="ns1:GoogleSearchResult">
<documentFiltering xsi:type="xsd:boolean">true</documentFiltering>
<estimatedTotalResultsCount
    xsi:type="xsd:int">32</estimatedTotalResultsCount>
…
<searchTime xsi:type="xsd:double">0.124633</searchTime>
<resultElements xmlns:ns3="http://schemas.xmlsoap.org/soap/encoding/"
    xsi:type="ns3:Array" ns3:arrayType="ns1:ResultElement[10]">
<item xsi:type="ns1:ResultElement">
…
```

5.3 Case Study – Web Services for Data Using the DataSpace Transfer Protocol (DSTP)

Data Space Transfer Protocol (DSTP)

- Designed to
  - Simplify working with remote data
  - Simplify working with distributed data
- Views data as geographically distributed collection of keys & attributes
- Works with data in
  - flat files
  - databases
  - proprietary formats (HDF, PDB data, etc.)
- Compliant with W3C web services & OGSA

DSTP: Key Concepts

- DataSpace - data distributed over a global computational grid or web
- Data organized into records with attributes
- Support for data & metadata
- Data can be integrated using distributed keys
  - Universal correlation keys - UCKs k[i]
  - correlate data (k[i], x[i]) on one server
  - with data (k[j], y[j]) on another server

Key DSTP Services

1. Data & metadata access (DSTP, SQL)
   - using XML metadata, range queries & sampling
2. Data transport (DSTP)
   - DSTP and XML/SOAP
3. Data integration by universal key
   - globally unique UCKs for joining distributed data
4. Data analysis and mining (PMML)
   - using algorithms for clustering, regression, etc.

1. DSTP Selects & Accesses Data

DSTP selects data:

- metadata browsing
- selecting attributes
- range queries on records
- SQL queries
- sampling …

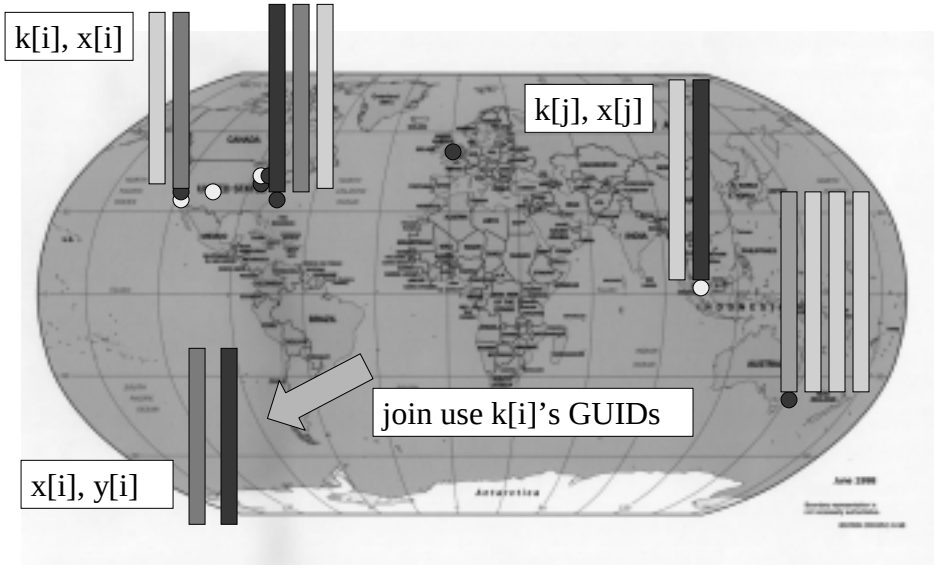
2. DSTP Transports Data

- DSTP servers support:
  - XML/SOAP services for transporting metadata & small data sets
  - streaming data transport services for data sets and their subsets
- Larger data and more complex data sets need specialized transport mechanisms as we will see in the next section

3. DSTP Integrates Data

- Each attribute x[j] is associated with a (distributed) key k[i]
- Each k[i] is associated with a GUID
- Two distributed attributes x[i] and y[j] which share the same k[i] can be integrated by the DSTP servers
- Works well in practice – much simpler than ontologies

UCKs Join Distributed Attributes



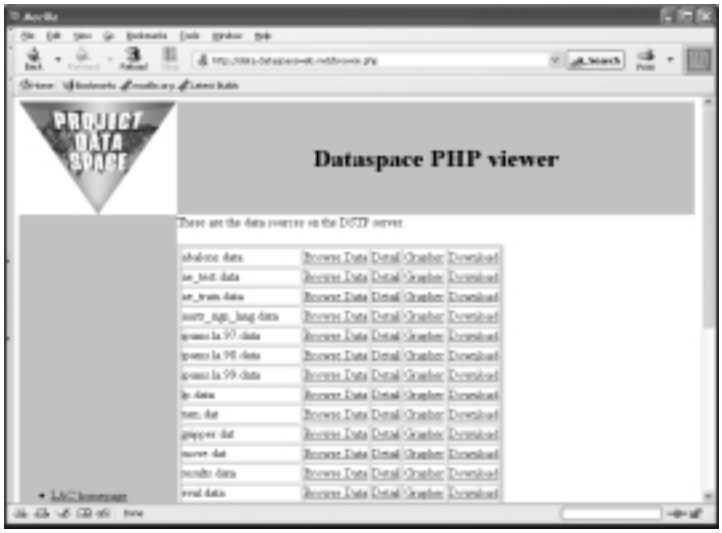
4. DSTP Supports Templated Data Mining Operations

- PMML is an XML mark up language for statistical and data mining models
- PMML has models for clustering, regression, association rules, neural networks, etc.
- DSTP requests can execute PMML supported data mining queries
- DSTP is not designed to support arbitrary grid like computations

Outline of Sample DSTP Session

- Use UDDI to discover DSTP server containing appropriate data
- DSTP client connects to DSTP server
- retrieve data set XML metadata using TCP
- select data set
- retrieve attribute metadata using TCP
- retrieve 25 columns of data and 20% subset of rows using SABUL/UDT

Exploring Data with DSTP

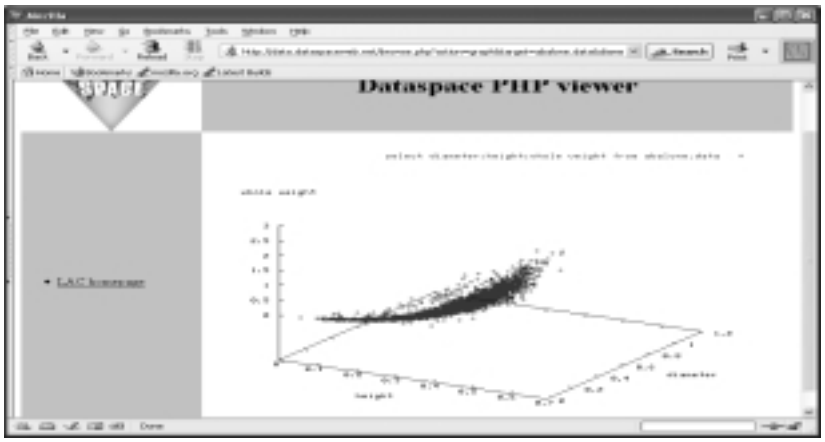


DSTP Supports Metadata

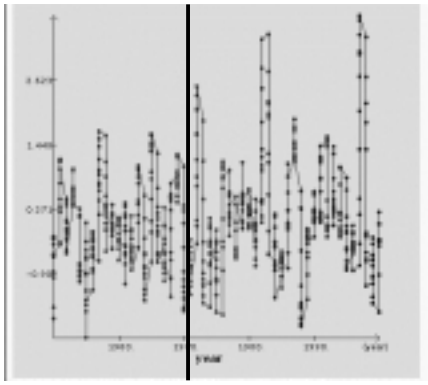
Browse or Download Data

Graph Data

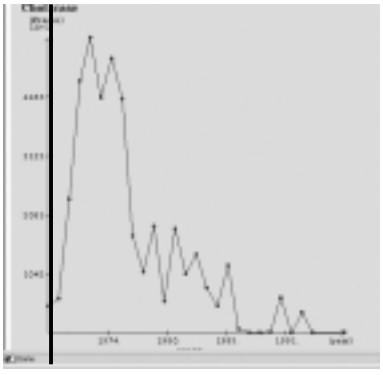
Graphing Data (cont'd)



Execute Templated Data Mining Operations



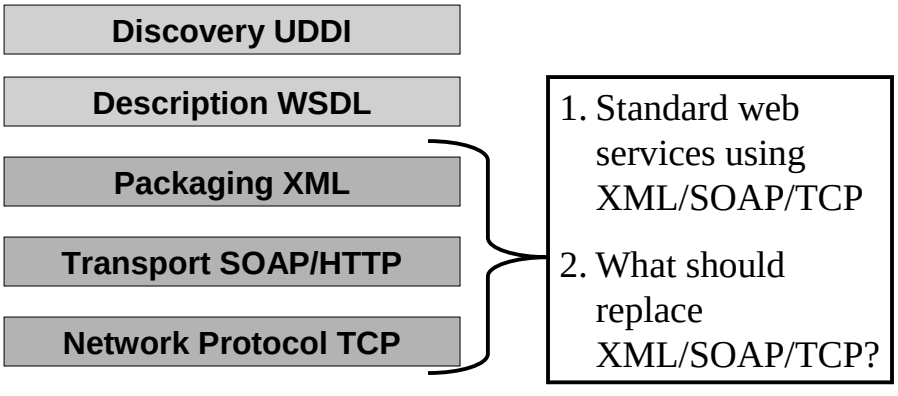
El Nino Data at NCAR



Cholera Data at WHO

5.4 High Performance Web Services

How to Scale Web Services?



Performance Issues with SOAP/XML

- SOAP/XML cannot transport binary data directly
  - Low efficiency for format conversion
  - Increased data size in ASCII format
  - Data may be encrypted and is not expected to be exposed to others
- Transport layer inefficiency
  - TCP is poor for bulk data transfer over long haul links

Three Methods to Scale SOAP/XML

1. Modify SOAP to include support for binary data
   - Direct Internet Message Encapsulation (DIME)
   - Not a standard now.
   - Does not solve the TCP inefficiency problem
2. Send data as MIME attachment
   - Header overhead/inconvenience/memory allocation problem/complexity

Three Methods to Scale SOAP/XML

3. Use SOAP for control and a separate channel for data
   - Pros:
     - Support multiple transport methods
     - Fast
     - Can use regular transport method if the negotiation to start a binary data channel fails
   - Cons:
     - Incompatibility with existing standard

High Performance Web Services using "SOAP+"

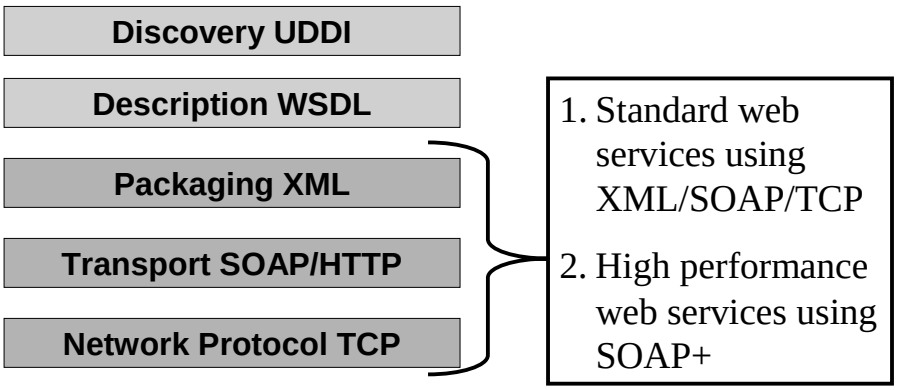
- Data can be in "streaming formats" in addition to XML
- Separate control and data channels:
  - Commands, control information, and metadata are transferred using SOAP/TCP.
  - Data can also be sent using a high performance data transport protocols over a separate channel

Streaming Formats for Data

- Compressed XML
- X-Scan: Process XML of the incoming data streams in real time
- SMIL: Synchronized Multimedia Integration Language
- DSTP: DataSpace Transfer Protocol (DSTP)

5.5 Case Study – High Performance Web Services for Data Using Open DMIX

Scaling Web Services with SOAP+



Open DMIX

- Standards based web services for data mining, data integration and data exploration
- Use SOAP/XML for control channel and small data
- Uses "SOAP+" for large data
  - streaming formats such as compressed XML or DSTP
  - high performance data transport such as SABUL/UDT
- Open source implementations available on Source Forge

Experimental Study 1 Using Open DMIX

- Web service request for geospatial data service in Amsterdam from a client in Chicago
  - < 5Mbps on a regular SOAP/XML/TCP architecture
  - 600Mbps (limited by disk write speed) on SOAP+ (TCP control; UDT data in DSTP format)

Experimental Study 2 Using Open DMIX

	SOAP+/DSTP	SOAP/XML
Select *	81 sec	634 sec
Range query	163 sec	302 sec

- medium data set - 10 million rows of data
- 320 MB
- Chicago to Amsterdam (110 ms RTT)

Experimental Study 3
Using Open DMIX

- Merge two data streams in real time by their UCKs
- 1 Gb/s link, 110 ms RTT
- Goal: process data at line speed by best effort algorithm

window size (records)	Random (%)	Match (%)	Speed (Mbps)
10000	2	92	600
10000	10	82	630
10000	20	79	655
10000	33	71	670

5.6 Summary & Conclusion

Summary

- More research is required to develop effective protocols and services for data grids, data webs & data mining grids
- We need high performance protocols both to
  - move files of bits and
  - collections of data records with attributes
- Web services emerging as common framework for several different approaches

Summary (cont'd)

- Active research with open source implementations for:
  - SOAP/XML
  - OGSA DAI
  - Open DMIX
- Initial work on developing high performance web services
  - SOAP for control and metadata
  - Use separate high performance data channel

For More Information

- LAC web site:
  - http://www.lac.uic.edu
- DataSpace Software
  - http://sourceforge.net/projects/dataspace

References

1. Robert Grossman, and Marco Mazzucco, DataSpace - A Web Infrastructure for the Exploratory Analysis and Mining of Data, IEEE Computing in Science and Engineering, July/August, 2002, pages, 44-51.
2. Web Service. http://www.w3.org/2002/ws/
3. Kenneth Chiu, Madhusudhan Govindaraju, Randall Bramley, Investigating the Limits of SOAP Performance for Scientific Computing, Indiana University. Accepted for publication in the Proceedings of HPDC 2002.
4. Simple Object Access Protocol. http://www.w3.org/TR/SOAP/.
5. R. L. Grossman, Y. Gu, D. Hanley, X. Hong, D. Lillethun, J. Levera, J. Mambretti, M. Mazzucco, and J. Weinberger, Experimental Studies Using Photonic Data Services at IGrid 2002, FGCS, 2003.
6. Marco Mazzucco, Asvin Ananthanarayan, Robert L. Grossman, Jorge Levera, and Gokulnath Bhagavantha Rao, Merging Multiple Data Streams on Common Keys over High Performance Networks, Proceedings of Supercomputing 2002, IEEE and ACM.
7. Robert van Engelen, Pushing the SOAP Envelop With Web Services for Scientific Computing, ICWS03.
8. Semantic Web. http://www.w3.org/2001/sw/.
9. Open Grid Service Architecture. http://www.globus.org/ogsa/.
10. Vijayshanker Raman, Inderpal Narang, Chris Crone, Laura Haas, Susan Malaika, Tina Mukai, Dan Wolfson and Chaitan Baru. Data Access and Management Services on Grid. www.cs.man.ac.uk/grid-db/papers/dams.pdf.
11. Hongsuda Tangmunarunkit, Stefan Decker, Carl Kesselman (University of Southern California): Ontology-based Resource Matching - The Grid meets the Semantic Web. http://www.isi.edu/~stefan/SemPGRID/proceedings/7.pdf.

References (cont.)

12. Shu-Ching Chen, Chengcui Zhang, Sheng-Tun Li, Hung-Chi Chen, Mei-Ling Shyu: Streaming SMIL Presentations via a Multimedia Semantic Model. JCIS 2002: 919-922.
13. A. Szalay, J. Gray, A. Thakar, P. Kuntz, T. Malik, J. Raddick, C. Stoughton. J. Vandenberg,"The SDSS SkyServer – Public Access to the Sloan Digital Sky Server Data," ACM SIGMOD 2002, MSR TR 2001 104.
14. "TerraService.NET: An Introduction to Web Services ," Tom Barclay, Jim Gray, Eric Strand, Steve Ekblad, Jeffrey Richter, MSR TR 2002-53, pp 13, June 2002 .
15. Z. Ives, A. Levy, and D. Weld. Efficient evaluation of regular path expressions on streaming XML data. Univ. of Washington Tech. Rep. CSE000502.

Chapter 6
Data Transport and OGSA

William (Bill) E. Allcock

Argonne National Laboratory

A Story of Evolution

- The concept of Grid Computing has been around since the 1960s
- Definition of Grid problem has been stable since original Globus Project proposal in 1995
  - Though we've gotten better at articulating it
- But the approach to its solution has evolved:
  - From APIs and custom protocols…
  - to standard protocols…
  - to Grid services (OGSA).
- Driven by experience implementing and deploying the Globus Toolkit, and building real applications with it

But Along The Way…

- Heterogeneous protocol base was hurting us
- Increasing number of virtual services that needed to be managed
- Moore's Law had time to work
- Network speeds increasing faster than Moore's Law
- Web services (WSDL, SOAP) appeared

Web Services

- At the heart of Web services is:
  - WSDL: Language for defining abstract service interfaces
  - SOAP (and friends): Binding from WSDL to bytes on the wire
- Web services appears to offer a fighting chance at ubiquity (unlike CORBA)
- But Web services does not go far enough to serve a common base for the Grid…

Transient Service Instances

- "Web services" address discovery & invocation of persistent services
  - Interface to persistent state of entire enterprise
- In Grids, must also support transient service instances, created/destroyed dynamically
  - Interfaces to the states of distributed activities
  - E.g. workflow, video conf., dist. data analysis, subscription
- Significant implications for how services are managed, named, discovered, and used
  - In fact, much of Grid is concerned with the management of service instances

Standard Interfaces & Behaviors:

- Naming and bindings
  - Every service instance has a unique name, from which can discover supported bindings
- Lifecycle
  - Service instances created by factories
  - Destroyed explicitly or via soft state
- Information model
  - Service data associated with Grid service instances, operations for accessing this info
  - Basis for service introspection, monitoring, discovery
- Notification
  - Interfaces for registering existence, and delivering notifications of changes to service data

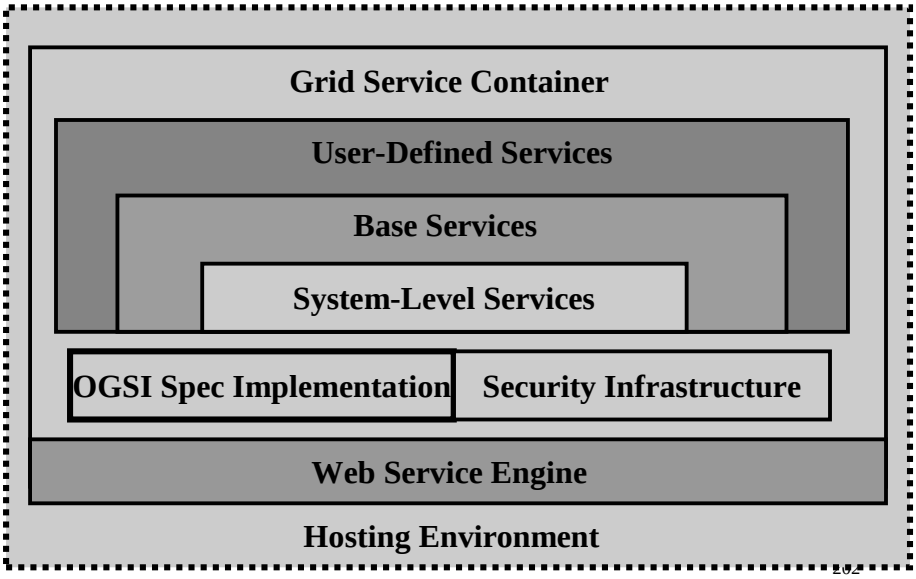
OGSI Grid Service Specification

- Defines WSDL conventions and GSDL extensions
  - For describing and structuring services
  - Working with W3C WSDL working group to drive GSDL extensions into WSDL
- Defines fundamental interfaces (using WSDL) and behaviors that define a Grid Service
  - A unifying framework for interoperability & establishment of total system properties

GT2 Evolution To GT3

- What happened to the GT2 key protocols?
  - Security: Adapting X.509 proxy certs to integrate with emerging WS standards
  - GRIP/LDAP: Abstractions integrated into OGSI as serviceData
  - GRAM: ManagedJobFactory and related service definitions
  - GridFTP: Unchanged in 3.0, but will evolve into OGSI-compliant service in 2004
- Also rendering collective services in terms of OGSI: RFT, RLS, etc.

GT-OGSA Grid Service Infrastructure

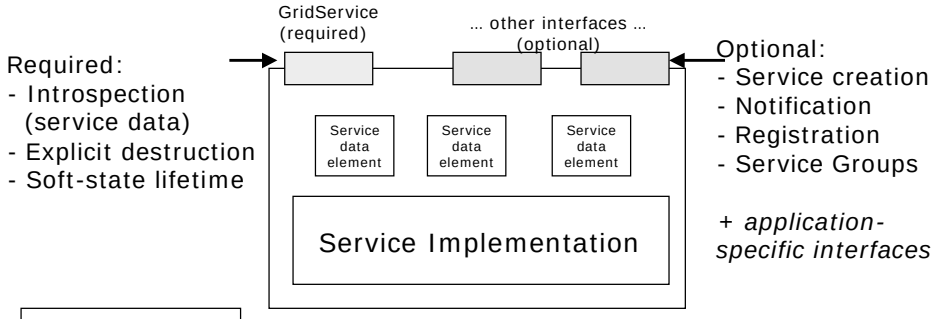


OGSI Core

GT3 Core:

OGSI Specification

The Specification Defines how Entities can Create, Discover and Interact with a Grid Service



GT3 Core:
OGSI Implementation

- GT3 includes a set of primitives that implement the interfaces and behaviors defined in the latest version of the OGSI Specification
- The implementation supports a declarative programming model in which GT3 users can compose OGSI-Compliant grid services by plugging the desired primitives into their implementation

GT3 Core:
OGSI Specification (cont.)

GridService portType

- Defines the fundamental behavior of a Grid Service
  - Introspection
  - Discovery
  - Soft State Lifetime Management
- Mandated by the Spec

GT3 Core:
OGSI Specification (cont.)

Factory portType

- Factories create services
- Factories are typically persistent services
- Factory is an optional OGSI interface

(Grid Services can also be instantiated by other mechanisms)

GT3 Core:
OGSI Specification (cont.)

Notification portTypes

- A subscription for notification causes the creation of a NotificationSubscription service
- NotificationSinks are not required to implement the GridService portType
- Notifications can be set on Service Data Elements
- Notification portTypes are optional

GT3 Core:
OGSI Specification (cont.)

Service group portTypes

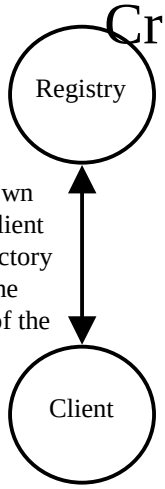
- A ServiceGroup is a grid service that maintains information about a group of other grid services
- The classic registry model can be implemented with the ServiceGroup portTypes
- A grid service can belong to more than one ServiceGroup
- Members of a ServiceGroup can be heterogeneous or homogenous
- Each entry in a service group can be represented as its own service
- Service group portTypes are optional OGSI interfaces

GT3 Core:
OGSI Specification (cont.)

HandleResolver portType

- Defines a means for resolving a GSH (Grid Service Handle) to a GSR (Grid Service Reference)
  - A GSH points to a Grid Service
    (GT3 uses a hostname-based GSH scheme)
  - A GSR specifies how to communicate with the Grid Service
    (GT3 currently supports SOAP over HTTP, so GSRs are in WSDL format)
- HandleResolver is an optional OGSI interface

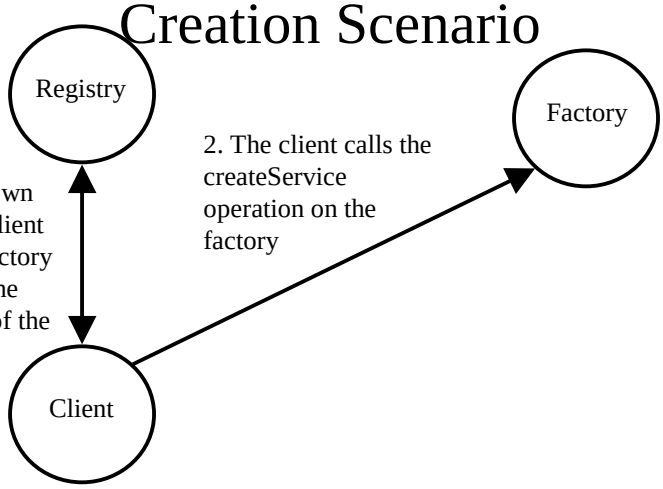
A Service Creation Scenario*



1. From a known registry, the client discovers a factory by querying the Service data of the registry

* The scenarios in this presentation are offered as examples and are not prescriptive

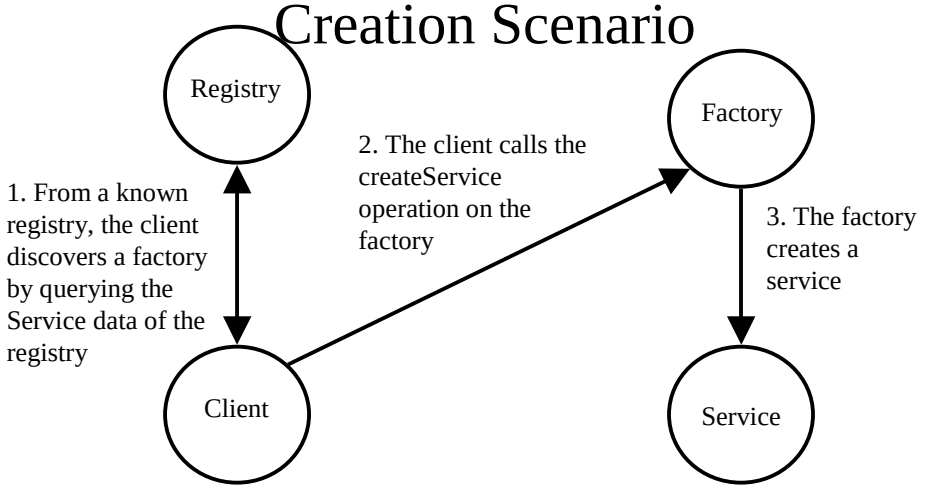
A Service Creation Scenario



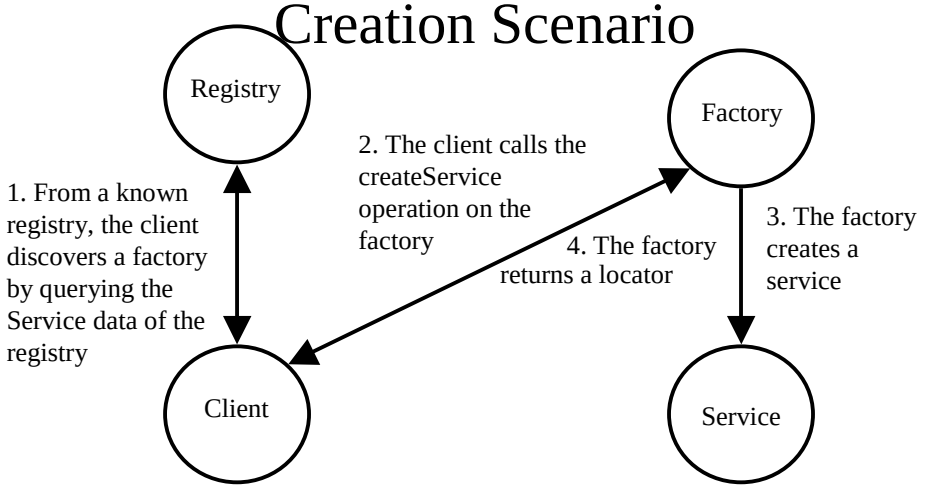
2. The client calls the createService operation on the factory

1. From a known registry, the client discovers a factory by querying the Service data of the registry

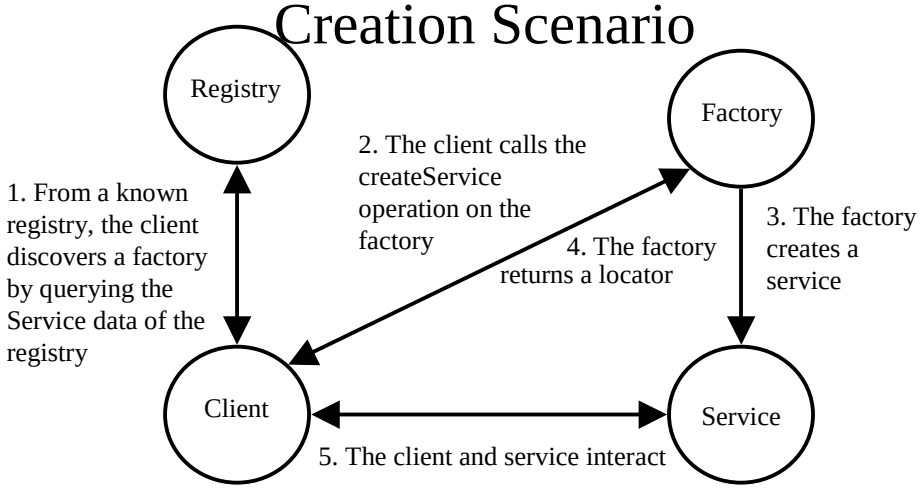
A Service Creation Scenario



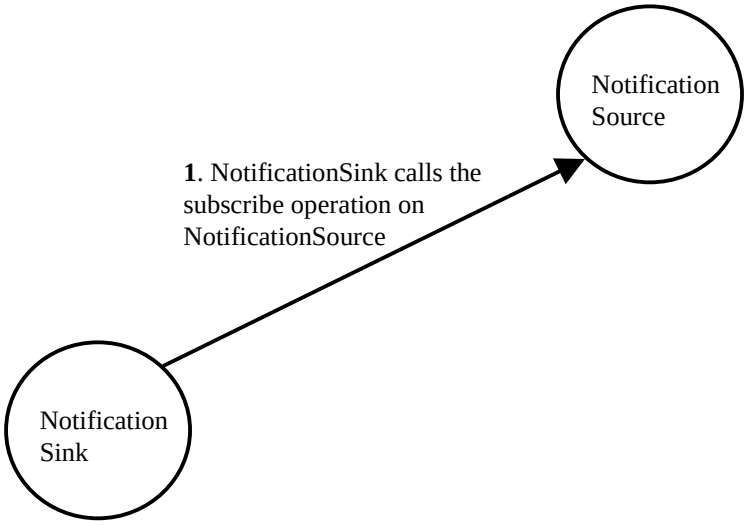
A Service Creation Scenario



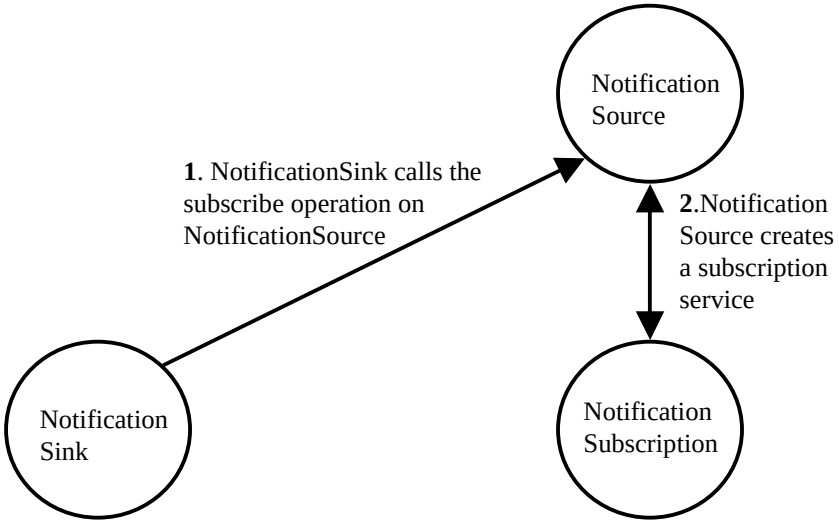
A Service Creation Scenario



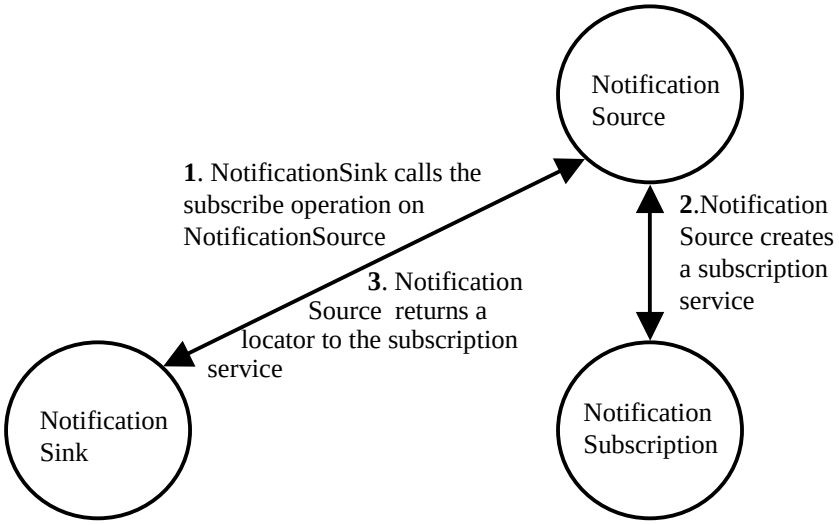
A Notification Scenario



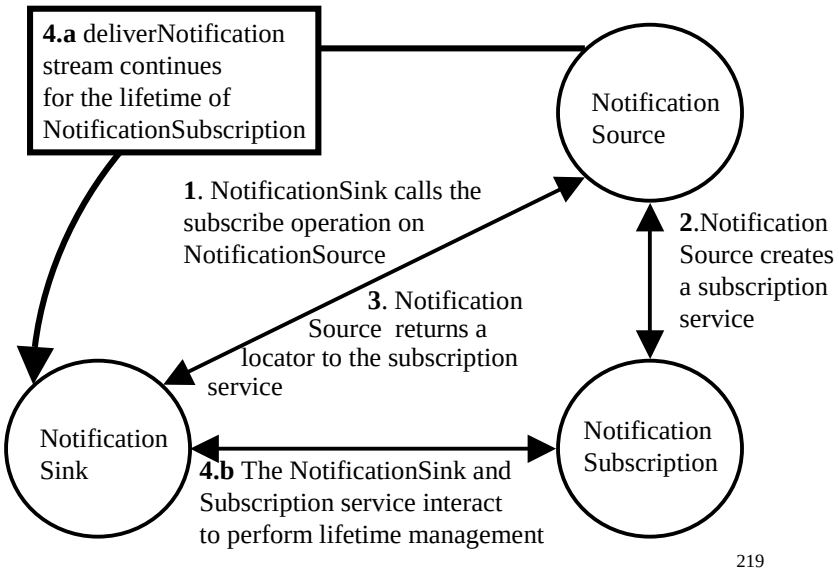
A Notification Scenario



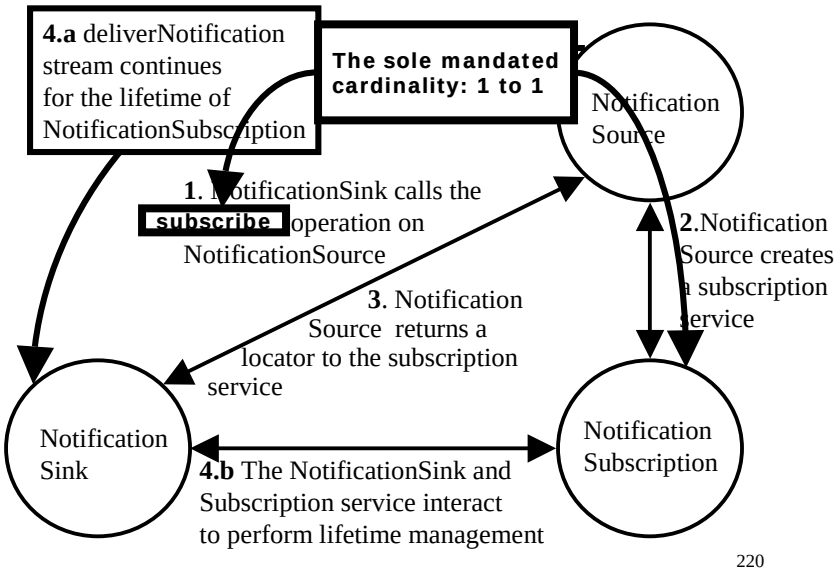
A Notification Scenario



A Notification Scenario



A Notification Scenario



Data Services in OGSA

Note: This is still evolving and will likely change. Tracking the GGF DAIS Working Group is the best way to stay current

Background

- The current GGF DAIS (Data Access and Integration Services) specification focuses on data access to databases
  - DAIS Goal**:** It must be possible to support existing un-modified data systems using the proposed interfaces through additional code
- The OGSA Data Services proposal (August 2003) has been produced in order to:
  - Incorporate DAIS requirements and general approach
  - Supports a broad, flexible, and extensible definition of "data service", beyond just the relational and XML database access interfaces that are being considered by DAIS (e.g. file systems, streams, devices, programs)
  - Exploit OGSI v1.0 (e.g. use service lifetimes to model client sessions rather than separate mechanisms)

Data Service Definitions [1]

- ***Data virtualization****:* An abstract view of some data, as defined by operations plus attributes (which define the data's structure in terms of the abstraction) implemented by a data service. ***Examples***: A file system, JPEG file, relational database, column of a relational table, random number generator
- ***Data interface (base):*** DataDescription, DataAccess, DataFactory, and DataManagement define mechanisms for inspecting, accessing, creating, and managing data virtualizations, respectively. They are expected to be extended to provide virtualization-specific interfaces.
  - An interface is a WSDL portType comprised of a set of *operations*

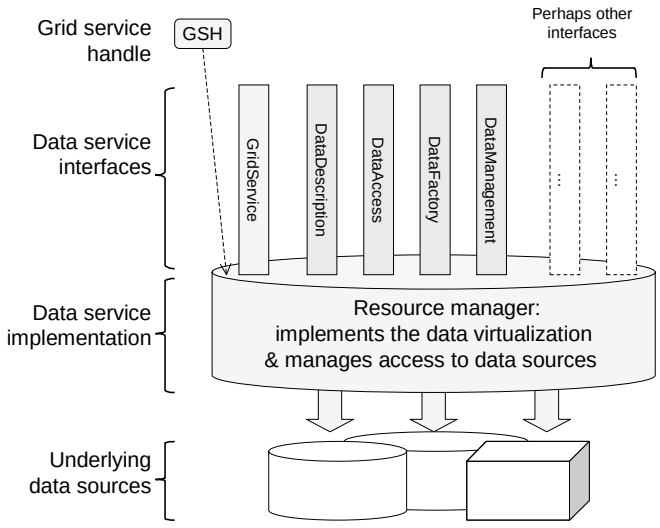
Data Service Definitions [2]

- ***Data service:*** An OGSI-compliant Web service that implements one or more of the four base data interfaces, either directly, or via an interface that extends one or more base data interfaces, and thus provides functionality for inspecting and manipulating a data virtualization.
- ***Data set*** : An encoding of data in a syntax suitable for externalization outside of a data service, for example for communication to/from a data service. ***Examples:*** WebRowSet XML, JPEG encoded byte array, ZIP encoded files

Data Service Definitions [3]

- ***Data source:*** A necessarily vague term that denotes the component(s) with which a data service's implementation interacts to implement operations on a data virtualization. ***Examples:*** A file, file system, directory, catalog, relational database, a sensor, a program.
- ***Resource manager***: The logic that brokers requests to underlying data source(s), via a data virtualization, through the data interfaces of a data service. ***Examples:*** An extension to, or wrapper around, a relational DBMS or file system; a specialized data service.
- ***DAIS-WG:*** GGF Working group that is producing the Data Access and Integration specification
- ***DAIS:*** Data Access and Integration Services specification

Data Service Overview



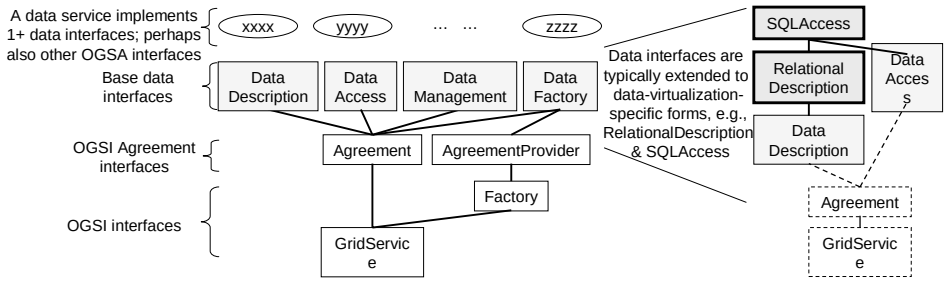
Base Data Service Interfaces [1]

- ***DataDescription:*** defines OGSI service data elements that describe the data virtualization supported by a particular data service
  - E.g. RelationalDescription, RowSetDescription, FileSystemDescription, FileDescription, JPEGDescription
- ***DataAccess:*** provides operations to access and modify the contents of a data service's data virtualization
  - E.g. SQLAccess, CursorRowSetAccess, StreamAccess, FileAccess, BlockAccess, TransferSourceAccess, TransferSinkAccess

Base Data Service Interfaces [2]

- ***DataFactory:*** supports a request to create a new data service whose data virtualization is derived from the data virtualization of the parent data service (the one that implements the DataFactory)
  - E.g. FileSelectionFactory, SQLFactory, TransferFactory, CollectionSelectionFactory
  - Some parallel the DataAccess specializations
- ***DataManagement*** : provides operations to manage the data virtualizations (and indirectly the data sources that underlie them) of a data service

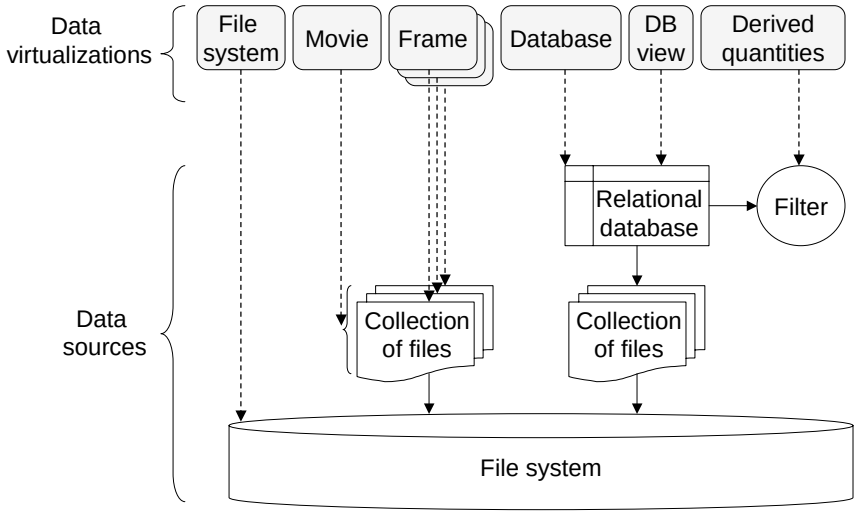
Interface Inheritance



Data Virtualization and Data Sources

- Flexible mappings between data virtualizations and underlying data sources and services. Examples:
  - **one-to-one**: A Data Service corresponds to a DB2 system instance that supports SQL.
  - **one-to-many**: A Data Service corresponds to a federated view of two or more underlying databases.
  - **many-to-one**: A Data Service offering XPath access to an XML File and SQL access to the same file though DB2 Data Federation.
  - **many-to-many**: Different views, each represented as a Data Service, of the one-to-many federation.

Multiple Virtualizations Example



Data Virtualization and Naming

- Each Data Virtualization (as a Grid Service) is represented by a GSH (Grid Service Handle)
- Each constituent data source has its own local namespace that describes the virtualization
  - Operations against a Data Service may use names (e.g., table names, file names) that can only be interpreted within the context of the service, in particular the data virtualization, to which the operation is directed.
  - If a global name is needed, you should use DataFactory to create a new virtualization (and thus GSH) that is appropriately scoped for your needs
- Data Virtualization implementation is responsible for directing requests to appropriate data sources.
  - Implementation = ***Resource Manager***

Data Virtualization and Service Lifetimes

- Data services can endure for either:
  - The lifetime of the Resource Manager
    - Example: To hold the data underlying the virtualization for the duration of the data service, independent of any particular clients. The associated ***DataFactory*** request may have the side effect of starting a resource manager such as a database system instance
  - The lifetime of the relationship between a resource manager and a set of clients (perhaps just one) interested in that data virtualization
    - Example 1: To create a virtualization containing a view of the parents' virtualization, to be shared with other clients
    - Example 2: To enable the processing of an SQL select where the result sequence is returned an item at a time

(OGSI-) WS-Agreement

- Recall key criteria of a Grid:
  - *Coordinates resources that are not subject to centralized control …*
  - using standard, open, general-purpose protocols and interfaces …
  - *to deliver non-trivial qualities of service.*
- Implies need to express and negotiate agreements that govern the delivery of services to clients
  - Agreement = what will be done, QoS, billing, compliance monitoring
- All interesting Web/Grid services interactions will be governed by agreements!

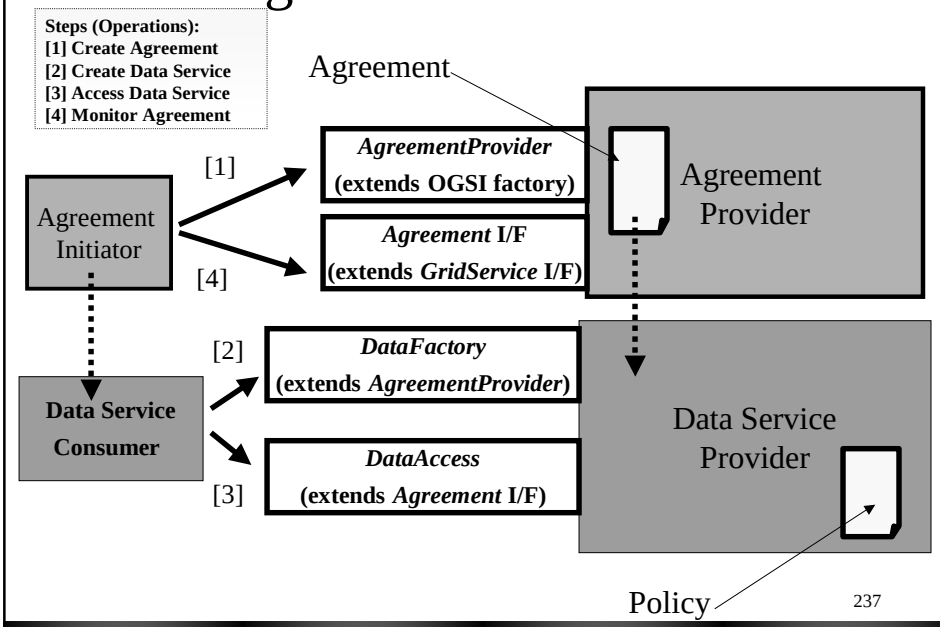
WS-Agreement Contents

- Standard agreement *language*
  - A composition of a set of terms that govern a service's behavior with respect to clients
  - Agreement language uses WS-Policy (currently)
  - Standard attributes for terms that express current state of negotiation
  - Other groups define specific terms
- Standard agreement negotiation *protocol*
  - Establish, monitor, re-negotiate agreement
  - Expressed using OGSI GWSDL interfaces
  - Each agreement represented by a service

WS-Agreement Interfaces

- ***AgreementProvider Interface***:
  - extends the OGSI Factory interface
  - defines how the Factory CreateService operation is used with the agreement language to instantiate an agreement with a service provider;
- ***Agreement Interface***:
  - extends the OGSI ***GridService*** interface:
    - The OGSI GridService interface provides operations for managing the lifetime of a service (and thus the agreement)
  - implemented by the service created by an ***AgreementProvider***
  - provides operations for the monitoring and re-negotiation of the terms of the agreement.

Agreement Overview



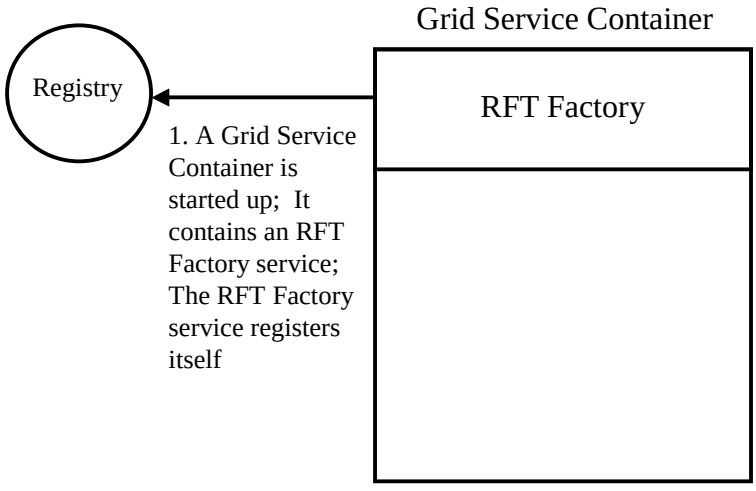
Agreement and Service Lifetimes

- Agreement Life Time
  - The agreement selection is made at data service create time.
  - The selected agreement can be redefined at any time within the scope of the selected agreement.
- Some Data Services (e.g. those associated closely with a Resource Manager) may have general agreements that apply to all clients, e.g.,
  - All data returned will be at most 5 minutes old
- Some Data Services may have individual agreements by client. They may be derived from some pre-defined base agreements, e.g.,
  - Platinum: 1 sec response time max
  - Gold: 5 sec response time max
  - Silver: 20 sec response time max

OGSI Compliant Transport Today

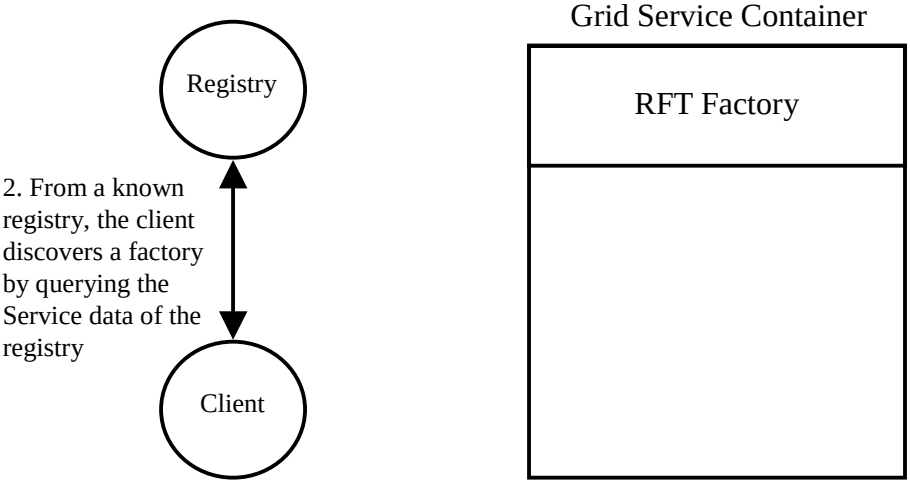
- Via the Reliable File Transfer Service
- Accepts a TransferRequest
  - SOAP Message
  - Defines Default transfer parameters such as TCP Buffer Size, parallelism, etc.
  - List of Source/Destination URL pairs
  - Defaults can be over-ridden per pair, if desired
  - URLs can be a directory and it will move the entire contents of the directory
- Service is OGSI compliant, executes a standard (non-OGSI compliant) 3rd Party GridFTP transfer

RFT in Action



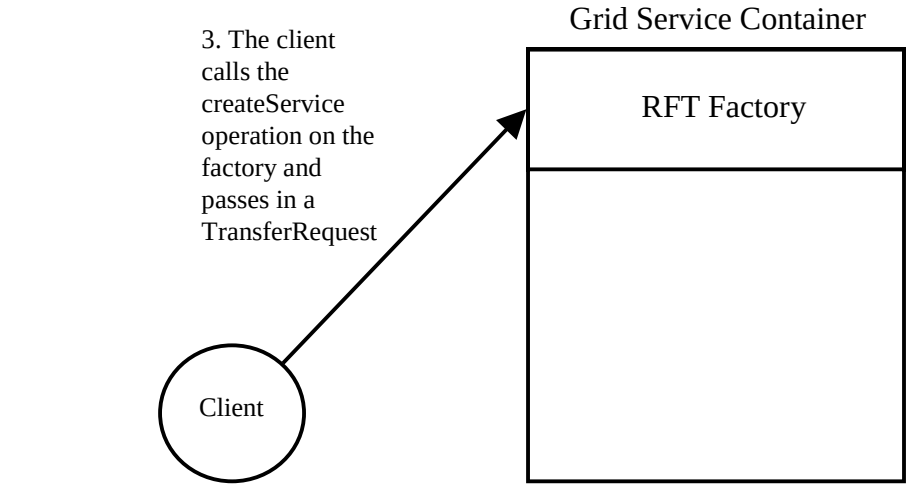
* The scenarios in this presentation are offered as examples and are not prescriptive

RFT in Action



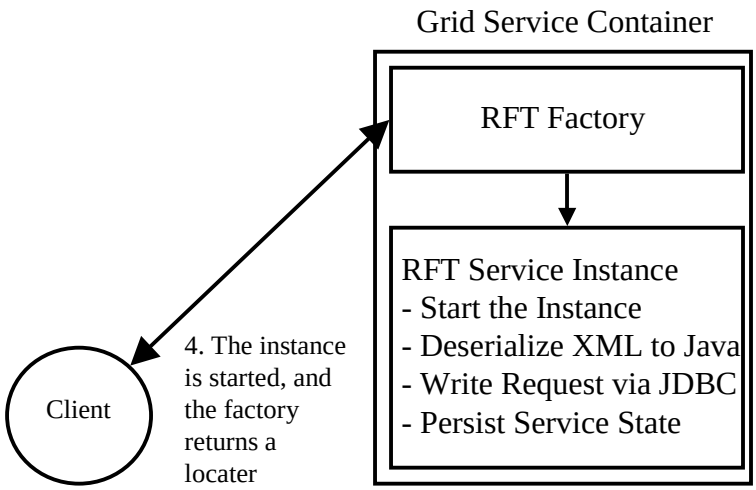
* The scenarios in this presentation are offered as examples and are not prescriptive

RFT in Action



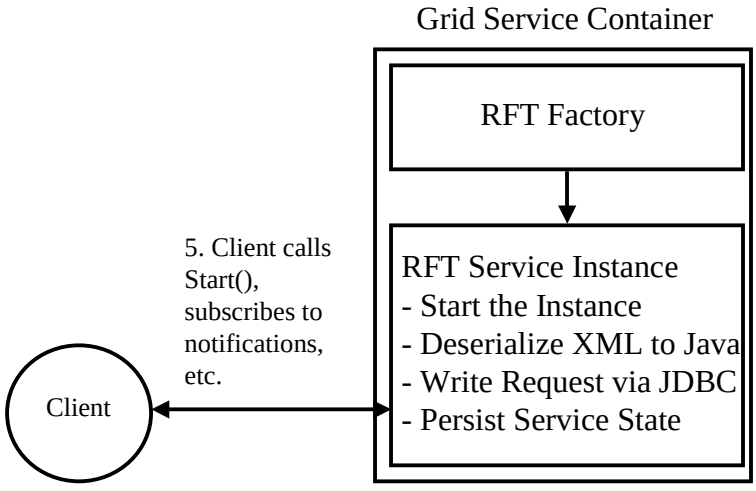
* The scenarios in this presentation are offered as examples and are not prescriptive

RFT in Action



* The scenarios in this presentation are offered as examples and are not prescriptive

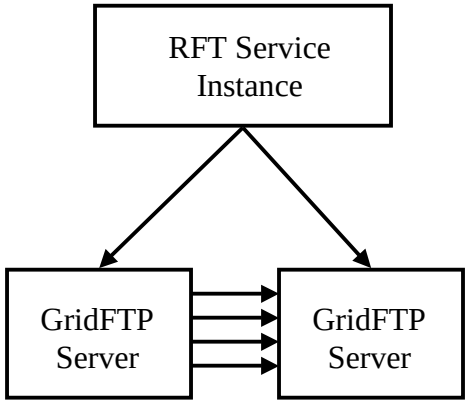
RFT in Action



* The scenarios in this presentation are offered as examples and are not prescriptive

RFT in Action

- Service is OGSI compliant
- Uses existing GridFTP (non-OGSI) protocols and tools to execute 3rd Party Transfer for the user
- Provides extensive state transition notification



* The scenarios in this presentation are offered as examples and are not prescriptive

OGSI Transport Tomorrow

- This is evolving and could change
- EVERYTHING will have a service interface.
- Transport will be negotiable
- Ideally, there will be autonegotiation based on proximity
  - Same process space: Shared memory
  - Same host: IPC
  - WAN: GridFTP

OGSI Transport

- Interface to transport services such as RFT will be via OGSI-Agreement
- ALL resources are represented as services. This includes files, file systems, databases, etc.
- Under the covers, likely that DataServices will have put() / get() interfaces on their DataAccess Interfaces.

Biographical Information for Presenters

William (Bill) E. Allcock

Bill Allcock is the technology coordinator and evangelist for GridFTP within the Globus Alliance. Bill has a BS in Computer Science and an MS in Paper Science. In his 15 years work experience he has been involved in a wide array of areas including computer networking, distributed systems, embedded systems, data acquisition, process engineering, control system tuning, and colloidal chemistry.

Bill's current research focus involves applications requiring access to large (Terabyte and Petabyte sized) data sets, so called DataGrid problems. He is also heavily involved in the Global Grid Forum. Bill has presented several previous tutorials on the Globus Toolkit(R), primarily on GridFTP use and development libraries. He also has lead tutorials on Introduction to Grids, as well as IO and security in the Globus Toolkit.

Robert Grossman

Robert Grossman is the Director of the Laboratory for Advanced Computing (LAC) and the National Center for Data Mining (NCDM) at the University of Illinois at Chicago (UIC). Robert Grossman became a faculty member at the University of Illinois at Chicago in 1988 and is currently Professor of Mathematics, Statistics, and Computer Science. He received a Ph.D. from Princeton in 1985 and a B.A. from Harvard in 1980. He is the Founder and President of Open Data Partners LLC, which provides consulting and outsourced data services. He is also the spokesperson for the Data Mining Group (DMG), an industry consortium responsible for the Predictive Model Markup Language (PMML), an XML language for data mining and predictive modeling, and the WS-DMX web services for data mining standard. He has published over 100 papers in refereed journals and proceedings.

Steven Wallace

Steven Wallace is the director of Indiana University's Advanced Management Lab. The lab specializes in network visualization, security, performance, and wireless management. Before creating the lab three years ago, Steven managed the engineering activities of the Abilene Network Operations Center as well as serving as a technical advisor to Indiana University's CIO. Steven has twenty year experience in software development and network management. Steven is a graduate of Indiana University