Alternate IPC Mechanisms

A Comparison of Their Use Within An ORB Framework

Chuck Abbott

Objective Interface Systems

Overview

- Rationale
- Goals
- Introduction
- Analysis
- Conclusions

Rationale

- All IPC Mechanisms are intra-host
- All presented OS's run on different machines
- Real-time and embedded applications have different run-time requirements
  - They range from very fast (embedded), to very predictable (hard RT)
- TCP/IP loopback has problems meeting these challenges
- Provide a mechanism to allow alternate "plug-in" transports to be used without changing application code
  - Take advantage of decreased latency and timeliness

Goals

- To examine other transport mechanisms as an alternative to TCP/IP loopback
- To compare the relative speed of the alternate transports to TCP/IP loopback
  - Look at average data transfer times from 8 to 32K bytes
- To observe the minimum and maximum deltas at a given point
  - Glimpse at predictability
- To compare the efficacy of using an alternate transport on a given operating system
  - Is it truly better than using TCP/IP loopback?
  - Are there OS specific behavioral differences?

© 2002 Objective Interface Systems, Inc.

Introduction 1

- **Tested two alternate transports**
  - Shared Memory
  - Message Queues (on one OS only)
- **All operating systems will remain anonymous**
  - Three desktop OS candidates (OS A, B, and C)
  - Two real-time OS candidates (OS D and E)
- **Compare transports on each operating system.**
- **Plug-in transports to a flexible ORB framework**
  - Sampled data at 10000 iterations on a dedicated host
  - No application code was changed for these tests

Introduction 2

- **TCP loopback**
  - Positive
    - Default ORB IIOP intra-host, inter-process communication mechanism
    - Very reliable, standard mechanism
    - Offers good location transparency
    - In most settings provides adequate performance
  - Negative
    - High latency
    - Low data through-put
    - Unpredictable data transfer rates
    - Widely varying Max delta for average data transfer times

Introduction 3

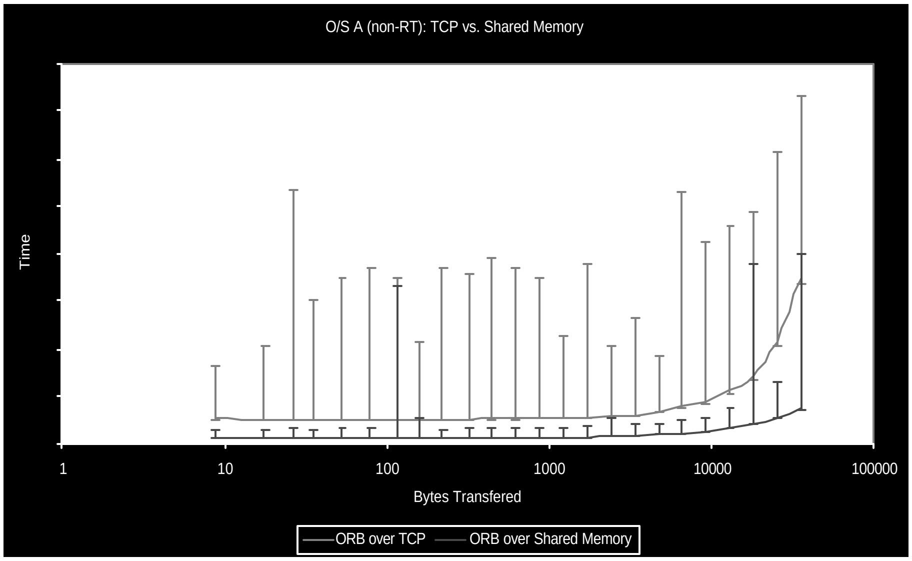
- **Alternatives to TCP/IP Loopback should provide**
  - Positive
    - Lower latency
    - Higher data through-put
    - More predictable data transfer rates
      - Although this may only happen on a well designed RTOS
  - Negative
    - Not specifically defined for IIOP
      - IIOP specifies GIOP over TCP
    - Potential loss of location transparency
      - The ORB might have to determine if the object exists on the same host
    - May not be widely available on all platforms

Analysis: OS A (Non RT) 1

- **TCP/IP loopback vs. Shared Memory**
  - Shared Memory decreases latency by a factor of
    - 4 for small data
    - 4.6 for large data
  - Shared Memory has a linear distribution
    - Suggests better scalability
  - Shared Memory has lower Maximum times

  - This OS shows considerable variability for both TCP and SM

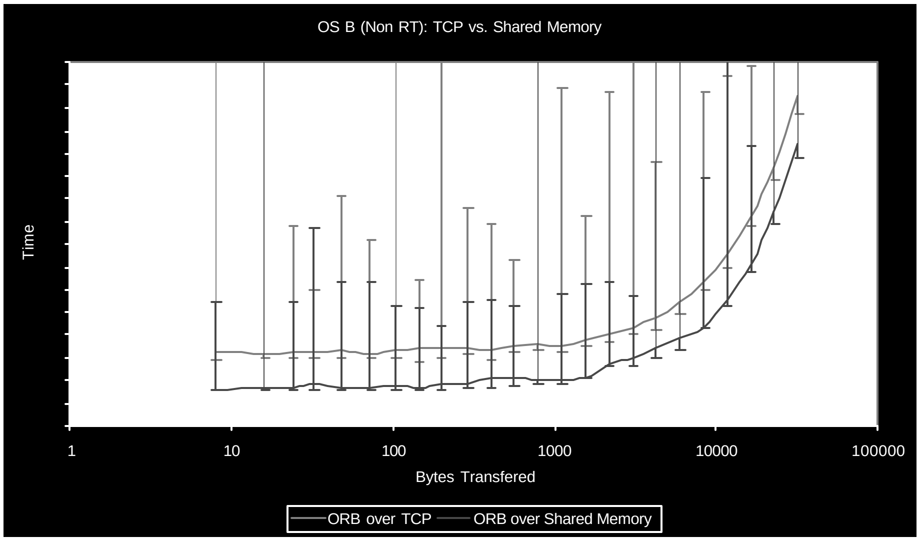
Analysis: OS A (Non RT) 2



Analysis: OS B (Non RT) 1

- **TCP/IP loopback vs. Shared Memory**
  - For small data < 1500 bytes SM decreases latency by a factor of 2
  - For larger data (32K) SM is nearly the same as TCP
  - Shared Memory scales about the same as TCP but is faster than TCP
  - Shared Memory and TCP share some fairly large max deltas
    - > 7000 usec
  - This OS shows considerable variability for both TCP and SM

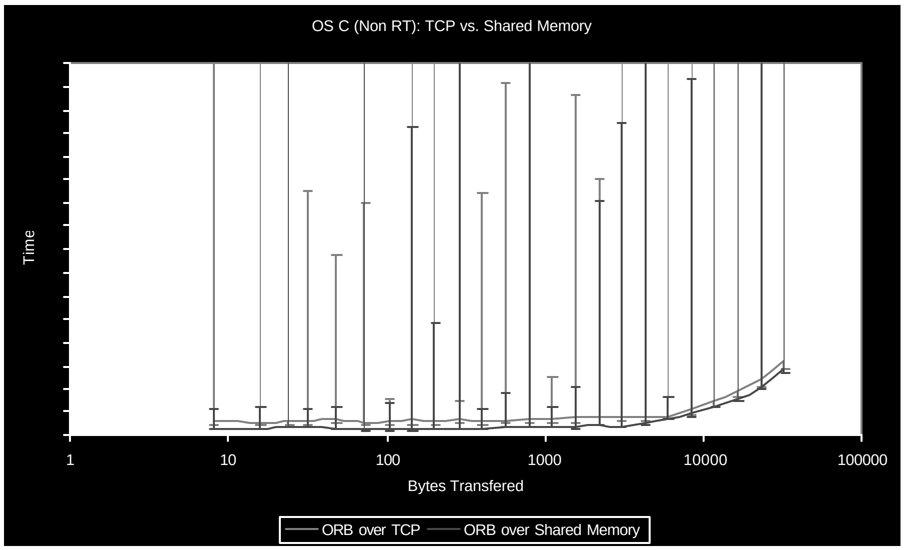
Analysis: OS B (Non RT) 2



Analysis: OS C (Non RT) 1

- **TCP/IP loopback vs. Shared Memory**
  - For small data < 1100 bytes SM decreases latency by a factor of 3 times
  - For larger data (32K) SM is nearly the same as TCP
  - Shared Memory scales a little better than TCP and is faster than TCP
  - Shared Memory and TCP share some fairly large max deltas
    - > 3000 usec
  - This OS shows considerable variability for both TCP and SM

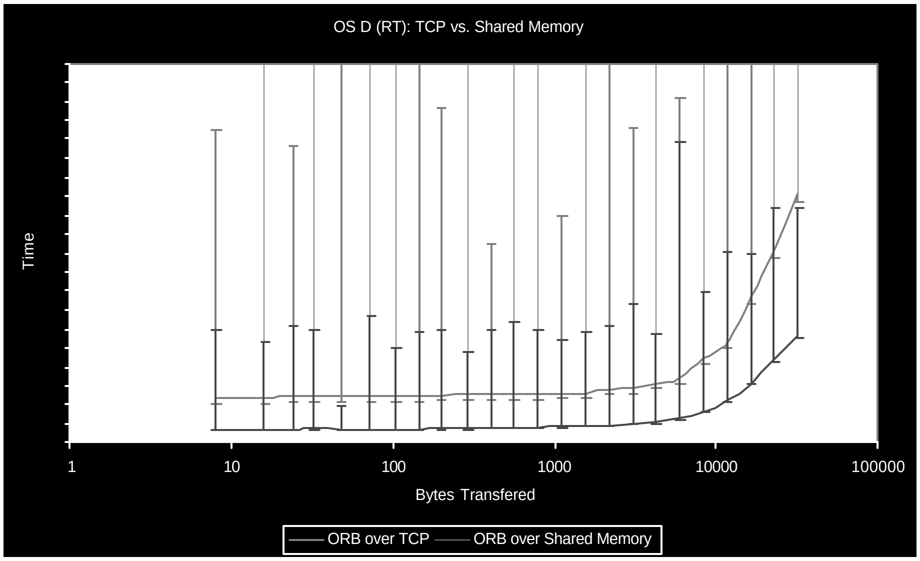
Analysis: OS C (Non RT) 2



Analysis: OS D (RT) 1

- **TCP/IP loopback vs. Shared Memory**
  - Shared Memory decreases latency by a factor of
    - 3.1 for small data
    - 2.3 for large data
  - Shared Memory has a linear distribution
    - Suggests better scalability than TCP
  - Shared Memory has fairly consistent Max deltas
    - Typically stays around 1500 to 2000 usec
- **TCP displays huge Max deltas (<25000 usec on first run)**
  - With other spikes between 2500 to 9500 usec
- **Shared Memory spikes highest at 4300 usec**

Analysis: OS D (RT) 2



Analysis: OS E (RT) 1

- **Generally any spikes seen are consistent**
  - 4000 - 4100 usec
  - Whatever the OS happens to be doing, the burp time is consistent
- **Transports Compared**
  - Message Queues
  - Shared Memory
  - TCP loopback
- **Surprisingly:**
  - Shared Memory was not a big performer
  - TCP wasn't so bad
  - Message Queues handily beat other transports for raw speed and scalability

Analysis: OS E (RT) 2

- **TCP/IP loopback vs. Shared Memory**
  - Shared Memory is about 20% faster for small data up about 1K
  - After about 2K, TCP is faster than Shared Memory
  - For large data (32K) TCP is 3.9 times faster than Shared Memory
  - All Spikes when they occur are about 4000 usec
  - The frequency of spikes is basically the same
  - TCP/IP loopback has a linear distribution
    - Suggests better scalability than Shared Memory

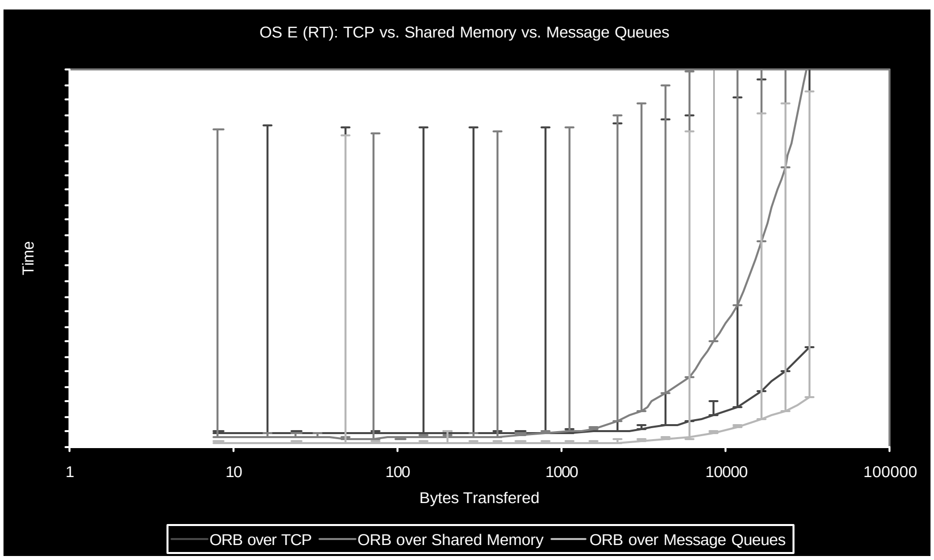
Analysis: OS E (RT) 3

- **TCP/IP loopback vs. Message Queues**
  - Message Queues decrease latency by a factor of
    - 3.8 for small data
    - 1.96 for large data
  - Consistently outperforms TCP
  - All Spikes when they occur are about 4000 usec
  - Spikes occur less frequently with Message Queues
    - Though they were starting to show up consistently for large data
      - As does TCP and Shared Memory
  - Message Queues has a linear distribution
    - Suggests better scalability than TCP/IP

Analysis: OS E (RT) 4

- **Shared Memory vs. Message Queues**
  - Message Queues decrease latency by a factor of
    - 3 for small data
    - 7.74 for large data
  - Consistently out performs Shared Memory
  - All Spikes when they occur are about 4000 usec
  - Spikes occur less frequently with Message Queues
    - Though they were starting to show consistently for large data
      - As does TCP and Shared Memory
  - Message Queues has a linear distribution
    - Suggests better scalability than Shared Memory

Analysis: OS E (RT) 5



Conclusions 1

- **Typically Shared Memory is a good bet over TCP/IP**
  - Lower latency
  - Greater through-put
- **Not all OS implementations of Shared Memory are equal**
- **Do not assume transport performance!!**
  - IPC Mechanisms can surprise you
    - Either faster or slower
  - For some OS implementations
    - Message Queues can be faster than Shared Memory
    - TCP/IP can be faster than Shared Memory

Conclusions 2

- **RTOS tend to to be more consistent**
  - Plot of average time is more linear
  - The ratio of Max to Min is smaller
  - There is more confidence that you can estimate a least upper bound
- **Given a potentially fast transport on a Non-Real-time OS**
  - You may decrease latency but not improve predictability
- **Given a potentially fast transport on a Real-time OS**
  - You may decrease latency and improve predictability

Conclusions 3

- **To create an optimal transport on any OS the transport designer needs to pay attention to**
  - Mutual exclusion devices
  - Data copy mechanisms
  - OS vendor optimizations
- **The application writer doesn't need to change the application based on the criteria above.**
- **For real-time systems an extensible transport framework is absolutely required**
  - Don't assume any one transport is the winner
  - Need the flexibility to change transports for changing environments

Thank You

chuck.abbott@ois.com

Also discussions on

www.realtime-corba.com