A demonstration of a real-time fault-tolerant distributed application

HPDC-6: August, 1997

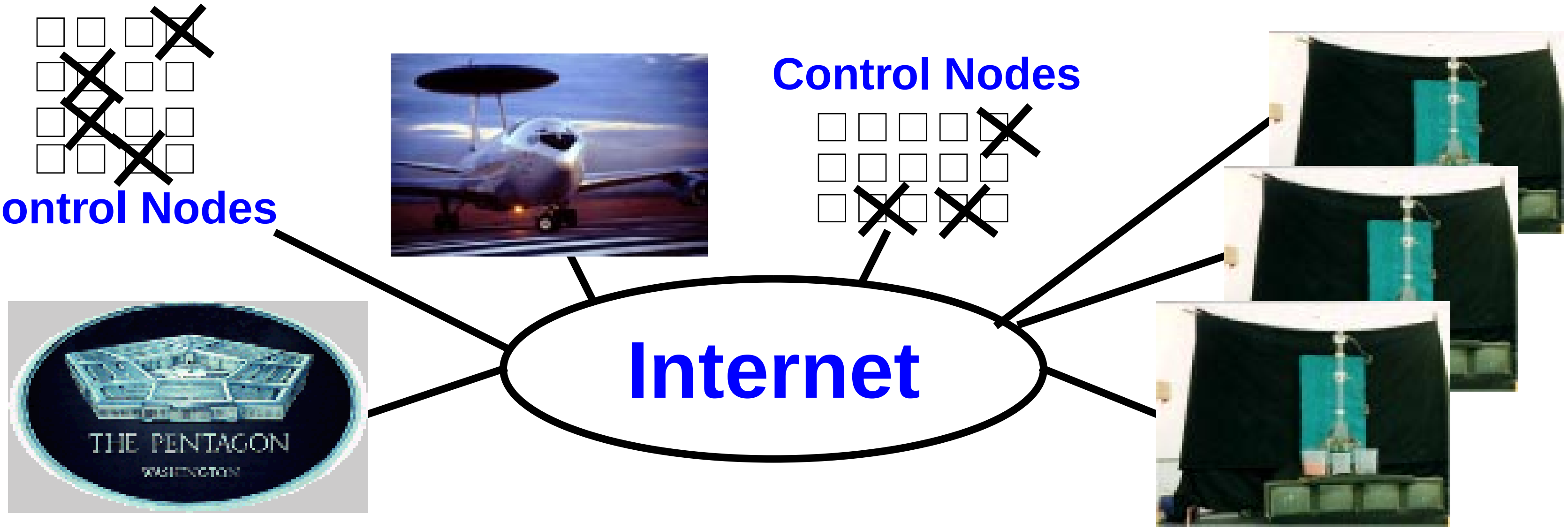
Douglas Wells
d.wells@opengroup.org

Laura Feeney
l.feeney@opengroup.org

The Open Group
Research Institute

http://www.opengroup.org/RI

Distributed Real-time Systems


Control Nodes

Control Nodes

Internet

Why bother with distributed real-time?

- Command and control problems are (unavoidably) distributed.
- Multimedia and collaborative (i.e. soft real-time) problems can be.
- Distributed solutions are scalable.
- Distributed systems have the opportunity to continue operation in the presence of partial failures.

Paradigms

The problem of unsorted golf balls...?

Virtual synchrony model ensures distributed consensus among nodes. Application can make powerful assumptions.

A communication framework provides end-to-end reservation and control of resources.

The result is predictable behavior in a dynamic (failure-prone) environment.

With a focus on configurability and adaptivity, these solutions are practical for problems that do not have hard real-time requirements.

MK 7

Trusted, distributed real-time, standards-compliant, operating system.

Single microkernel source base with scalable configuration options.

In real-time configurations:

- preemptible kernel
- real-time threads and synchronizers
- scheduling framework: separate policy and mechanism
- resource reservation (e.g. buffers)
- low level system clock management

CORDS: Communication Objects for Real-time Dependable Systems

Framework and toolkit for writing protocols and composing the stack from protocol graphs. Graphs can be instantiated in-kernel, as middleware, or both.

- Uniform interface and common utilities for all protocols
- Derived from the x-kernel (U. of Arizona)

Key addition is the “paths” abstraction, providing control over system resources, such as memory, CPU and bandwidth.

Within CORDS, we have implemented:

- KKT: a communication subsystem for multicomputer clusters
- LTS: a bounded-offset clock synchronization facility
- GIPC: process groups (strong membership, atomic bcast)

LTS: Bounded Offset Synchronization

Based on probabilistic algorithm due to F. Cristian et. al.

Provides a distributed counter with a guaranteed bounded offset from the master counter.

This distributed clock can explicitly fail with probability > 0.

LTS assumes a bounded drift of the local clock. LTS does not guarantee the frequency of the distributed clock.

By contrast, NTP uses statistical analysis to correct the local clock frequency based on input from other clocks.

Implemented as an MK kernel-level CORDS protocol and relies on its real-time properties.

GIPC: Real-time process group

GIPC group communication maintains consensus on membership and ordering of all messages.

Though simple API’s, GIPC provides an application with:

- Consistent view of group membership with sequence number
- FIFO atomic broadcast with sequence number
- Unreliable broadcast

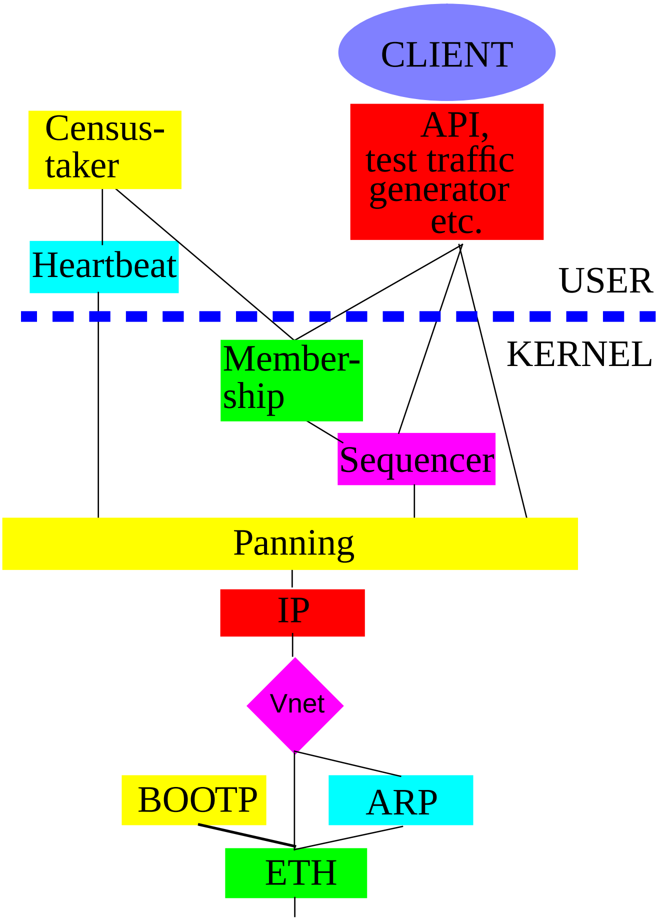
These are powerful abstractions for distributed computing.

Implementation within CORDS gives real-time properties:

- Predictable execution time
- System resource reservation

GIPC (cont'd)

A CORDS implementation of the GIPC service (MK)


CLIENT

API, test traffic generator etc.

Census-taker

Heartbeat

USER

KERNEL

Member-ship

Sequencer

Panning

IP

Vnet

BOOTP

ARP

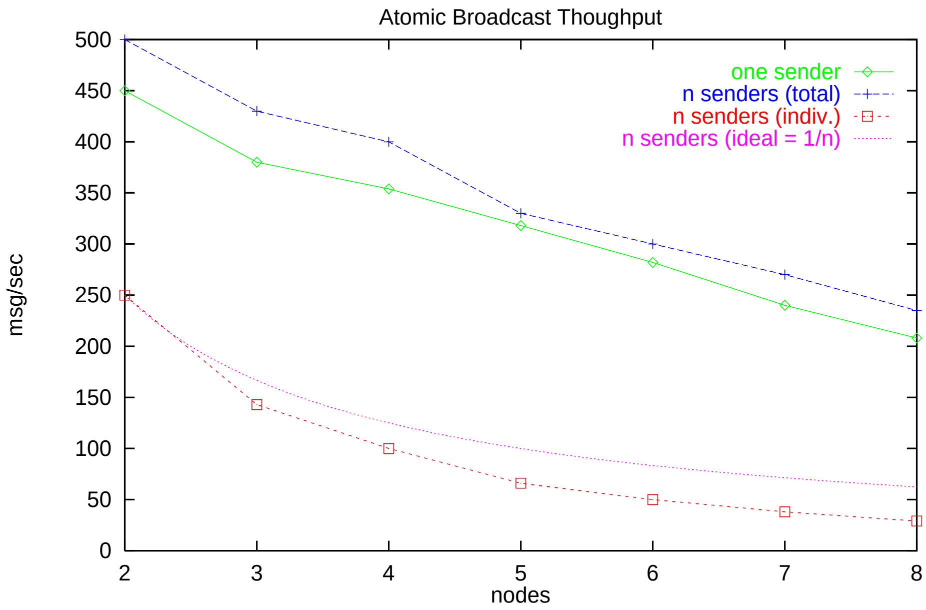
ETH

Some *preliminary* data

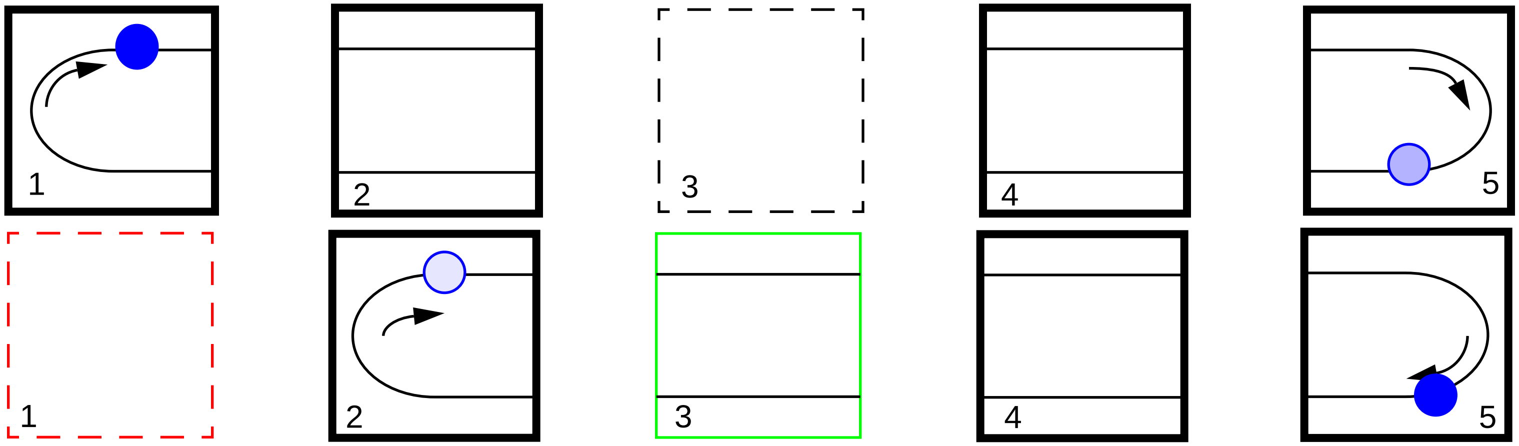
MK latency:

5 nodes, no node failures:
8.7 / 6.0 / 16 ms (mean/ min/ max)

Thoughput on HP-UX



Distributed Application: GIPC Ring demo



Visual demonstration of virtual synchrony and global state: All nodes must agree on membership and order of all traffic.

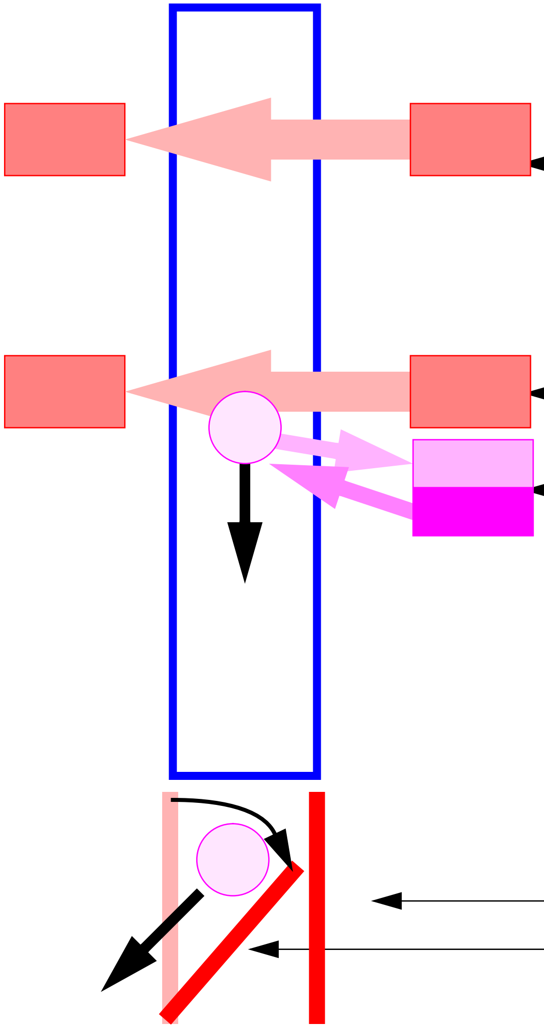
The ring immediately reconfigures on a membership change.

Node that owns the marker abcasts its position for 9 steps. Node controlling opposite position draws anti-marker in response.

When marker is about to transition, “next” node begins broadcasting.

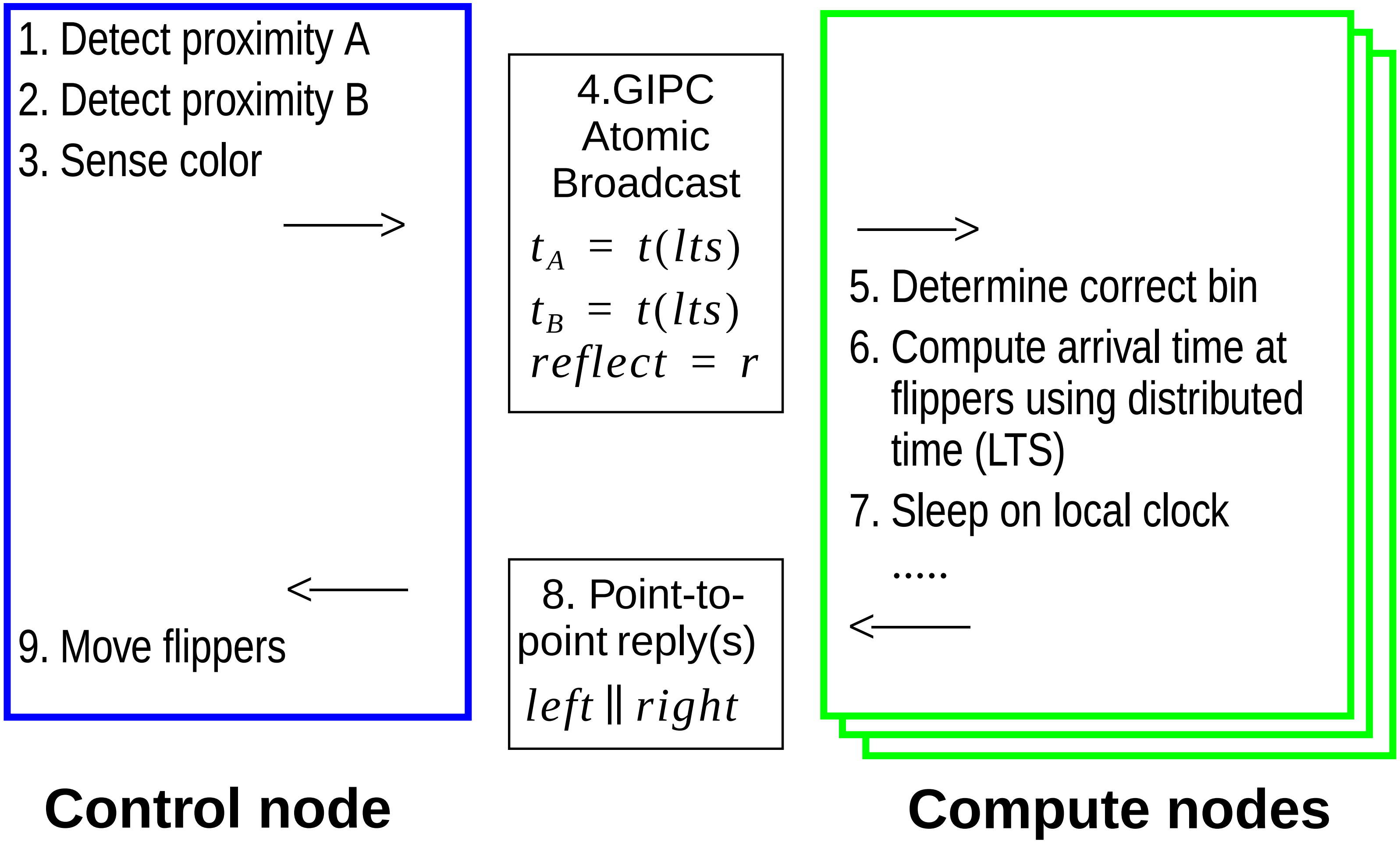
If marker is lost, anti-marker becomes marker.

Control Application: Ballsorting



Sensors	Control node
Proximity A	t_A
Proximity B	t_B
Color Sensor	$reflectivty\ =\ r$
Flippers	$\left(\frac{d^2x}{dt^2} = g\right) \Rightarrow t_{flip}$ $f(r) \Rightarrow left \Vert right$

Distributed Sorting Algorithm



1. Detect proximity A
2. Detect proximity B
3. Sense color

4.GIPC Atomic Broadcast

$t_A = t(lts)$

$t_B = t(lts)$

$reflect = r$

5. Determine correct bin
6. Compute arrival time at flippers using distributed time (LTS)
7. Sleep on local clock

.....

8. Point-to-point reply(s)

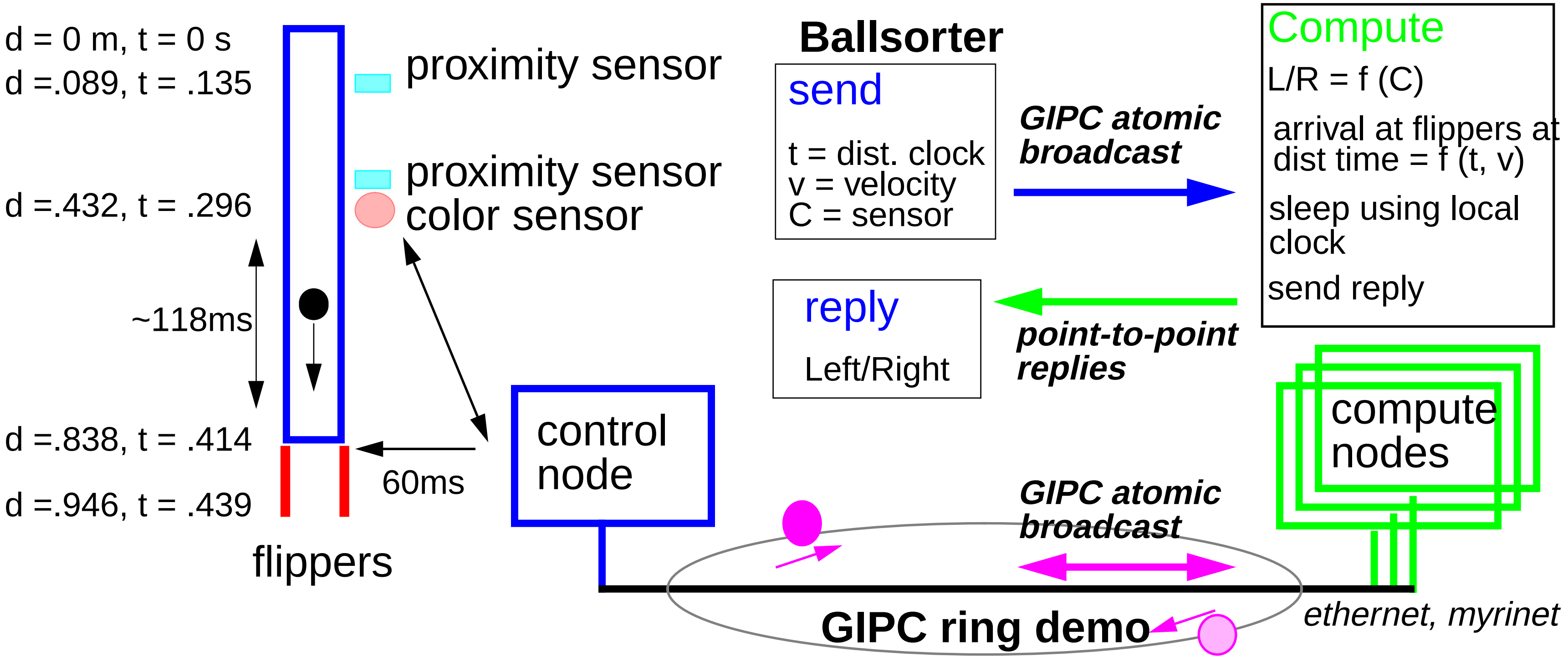
$left \| right$

9. Move flippers

Control node

Compute nodes

Distributed Ballsorter



LTS Clock synchronization: +/- 1.25ms, guaranteed bound

GIPC group membership change: up to ~250ms

Total sleep time: ~48ms

At 4 balls/sec, drop at most one ball per *compute* node failure

Impact and Future Work

CORDS and GIPC are extractable technologies:

- HP-UX and NT ports are used in our Scalable, Highly-available Web Server.
- DASCOM has announced use of CORDS for secure routers
- Alacron/Honeywell AVIS system uses CORDS over Myrinet
- U. Michigan RTCL/Honeywell Technology Center (ARMADA)
- U. Arizona (Prof. Schlichting)

Ongoing work to characterize and improve real-time performance -- sort balls under arbitrary load.

Next-generation CORDS will focus on dynamic protocol graphs, mobile code technologies for active networks, security and distributed resource management.

Experimental results: Paths

Overall udp/ip latency is comparable to commercial systems.

The use of paths reduces “jitter” of the udp/ip roundtrip times in the presence of network load.

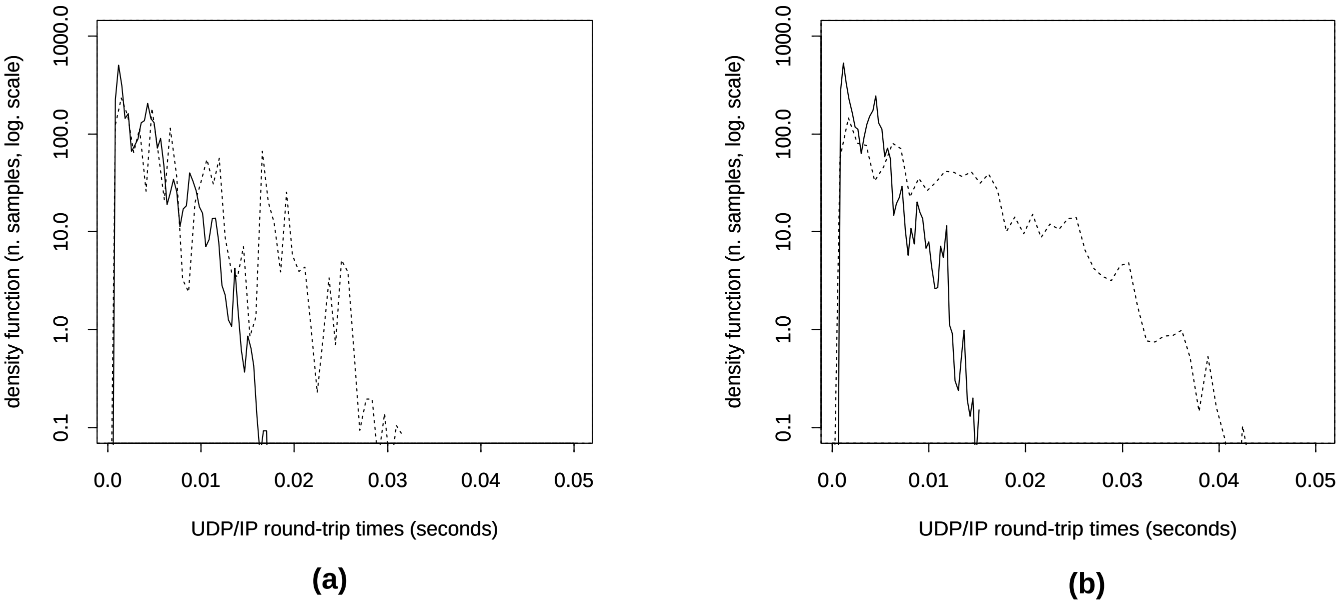
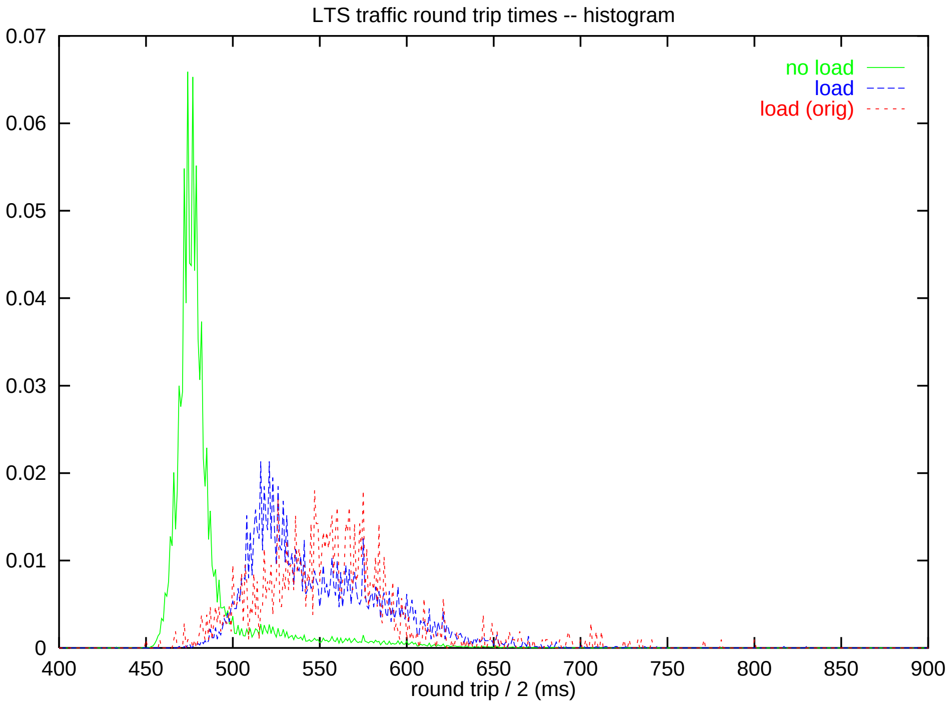


Figure 4: Density functions of UDP/IP round-trip times: *the use of paths with privileged resources (solid lines) limits Interference with simulated background traffic (two (a) and four (b) concurrent 16K memory to memory transfers initiated and sustained from both nodes).*

Experimental Results: LTS (preliminary)

Eventual goal is to sort balls under arbitrary (network and CPU) system load.

LTS depends on low “jitter” for LTS network traffic. Quantify the hit that LTS takes in the presence of hgh CPU load.



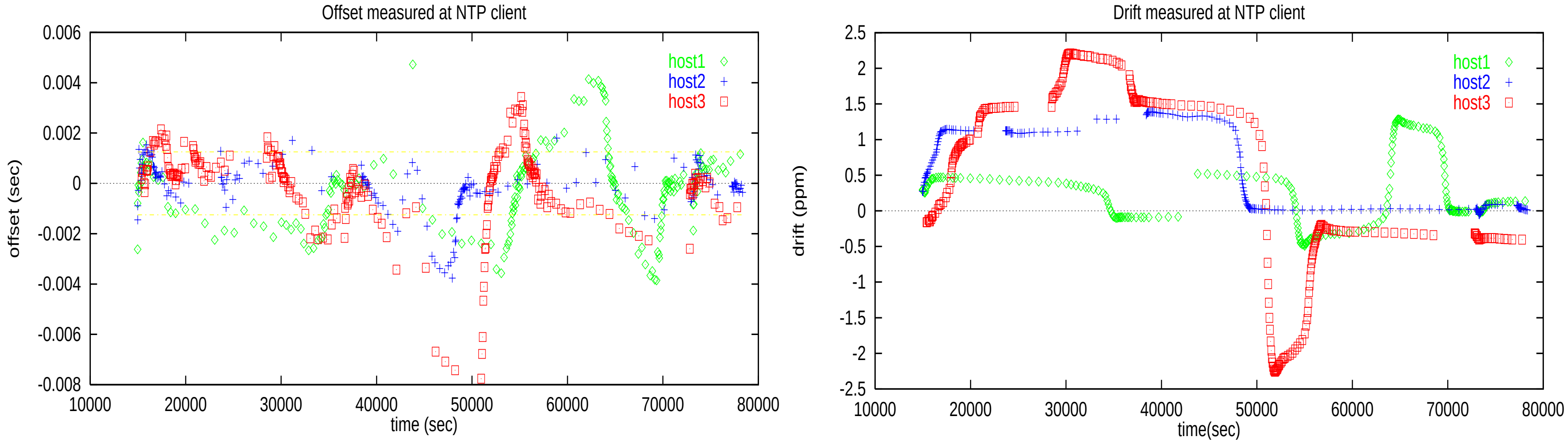
Experimental Results: LTS and NTP(prelim.)

LTS configuration and performance: 1ms bound

Configuration	Performance
assumed max drift: 200 ppm	rejected replies: 9.6%
retry period: 1.5 sec	rapport interval: 558 / 250 / 804 ms (avg/min/max)
roundtrip time (min/max): 800/1100 usec	roundtrip (valid replies): 976 ms avg.

NTP has low network cost, but gives no guarantees, even good conditions.

NTP Perfomance Data (loopstats)


Offset measured at NTP client
offset (sec)
time (sec)
host1 ◇
host2 +
host3 □
Drift measured at NTP client
drift (ppm)
time(sec)
host1 ◇
host2 +
host3 □