Simple Fairness Protocols for Daisy Chain Interconnects

Wladek Olesinski, Hans Eberle

Nils Gura

Sun Microsystems Laboratories

Bob Dickson, Aron Silverton

Sumti Jairath, Peter Yakutis

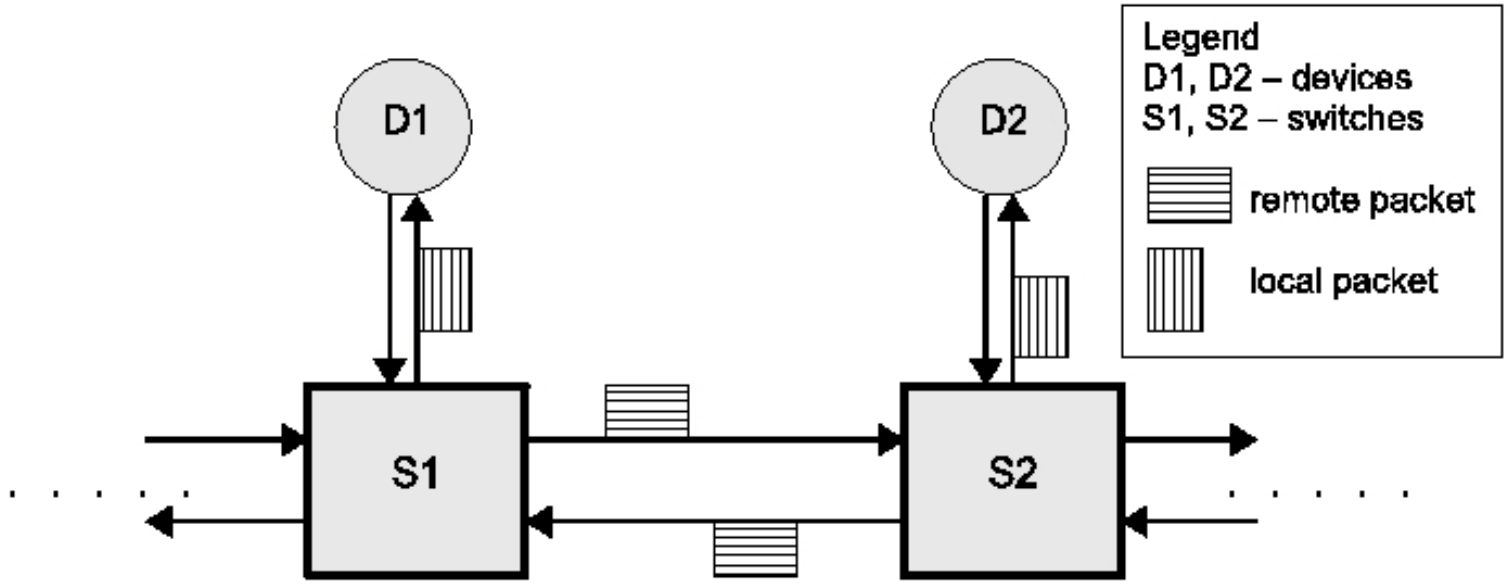
Sun Microsystems

Outline

- The problem
- Analysis of HyperTransport fairness protocol
- Our protocol and its properties
- Conclusions and future work

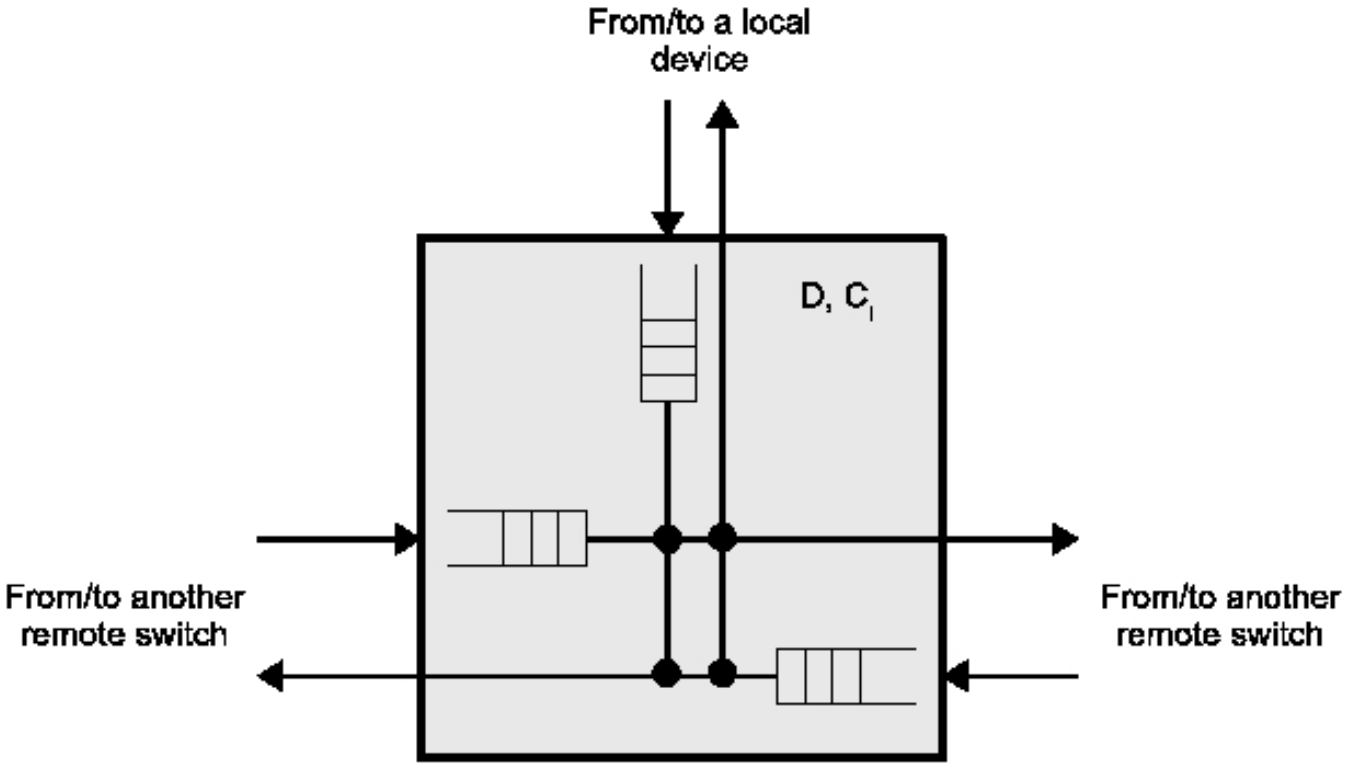
The Problem

How to provide fair access to bandwidth by local and remote traffic in a chain interconnect?



Switches in Daisy Chain Interconnect

- Bidirectional chain of links that connect simple switches

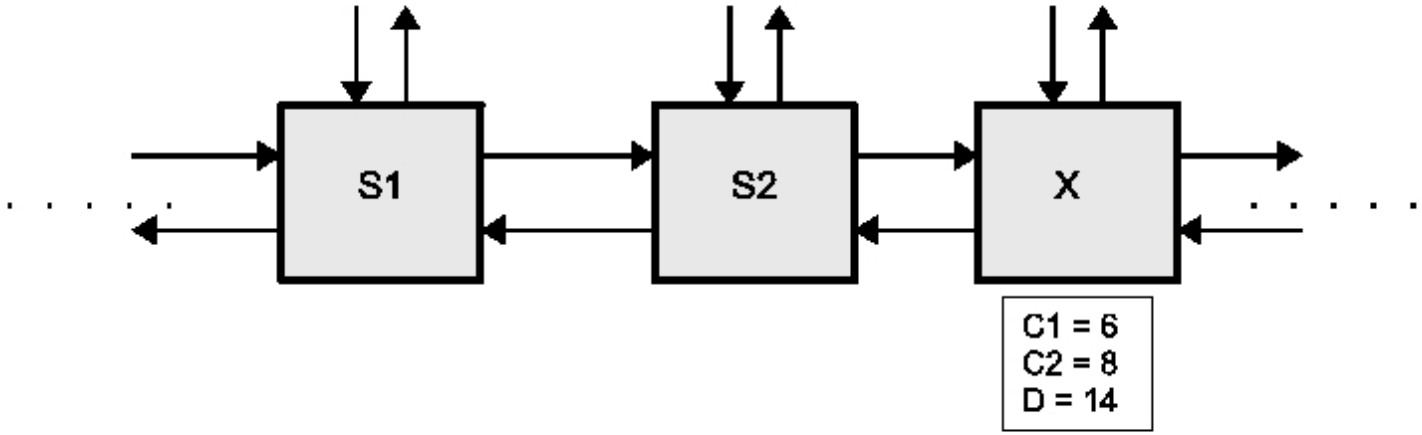


A Core of a Fairness Protocol

- Assume a chain with *N* switches. Every switch has:
  - Counter *D* incremented every time a remote packet is forwarded
  - *N x* counters C_i; counter C_i is incremented every time a packet from upstream switch S_i is forwarded
- When one of counters C_i reaches 8, the fairness algorithm calculates a new transmission rate of local packets
- How this rate is calculated differs between protocols

Example

- C_1=6 because switch *X* has seen 6 packets from S_1
- C_2=8 because switch *X* has seen 8 packets from S_2
- Counter *D=14*, calculation of transmission rate is triggered



Ideal Fairness Protocol *HTideal*

- Insertion rate in ideal fairness protocol *HTideal* is
  - $R_{ins} = 8 / D$
    - transmission rate $R = 8 / (D + 8)$
- Observation
  - If the upstream traffic is heavy, *D* is large
  - If *D* is large, insertion rate at a downstream switch drops
  - In fact, *R* is a transmission rate of the most active upstream source
- Example
  - For *D*=14, transmission rate at switch *X* is *R*=8/22=4/11
  - Pattern of remote and local packets: RRLRRLRRLRL

Disadvantages of *HTideal*

- *HTideal* is fair but:
  - Dividing numbers, out of which denominator is not power of 2, is expensive in hardware
  - The algorithm does not specify how remote and local packets are spaced to avoid bursts

HyperTransport Fairness Protocol *HT*

- Insertion rate in HyperTransport *HT* protocol is
  - $R_{ins} = 1 / r$, where $r = (D + n_{rand}) >> 3$
    - transmission rate: $R=1/(r+1)$
  - In other words, to get *r*, the protocol adds a random number between 0 and 7 to *D*, and divides it by 8
  - Notice similarity to ideal insertion rate $R_{ins} = 8 / D$
- Observations
  - A shift by 3 (division by 8) simplifies calculations but introduces rounding inaccuracy
  - Adding a random number is supposed to smooth out the results

HyperTransport Fairness Protocol *HT*, cont'd

- Example (for $D=14$)
  - If n_{rand} is 0 or 1, then $r=(14+n_{rand})>>3=1$, and transmission rate R=1/2
    - Pattern: RL
  - If n_{rand} is 2...7, then $r=2$, and transmission rate R=1/3
    - Pattern: RRL
- The rate varies depending on the random number

Example Unfairness of *HT*

- Example with *D=14* continued
- With probability 1/4, transmission rate *R=1/2*
  - Pattern: RL x 14
- After this sequence, a *new* transmission rate is calculated
- With probability 3/4, transmission rate *R=1/3*
  - Pattern: RRL x 7
- In every 4 periods in which *D=14*, on average, there is 1 RL x 14 sequence, and 3 RRL x 7 sequences
- This gives 14+42=56 remote, and 14+21=35 local packets → rate R=35/(35+56)=5/13

Example Unfairness of *HT*, cont'd

- Recall that the ideal transmission rate of *HTideal* is *R1=4/11*
- Local transmission rate of *HT* is *R2=5/13*
- Define *unfairness factor* as

$$f=(R2-R1)/R1$$

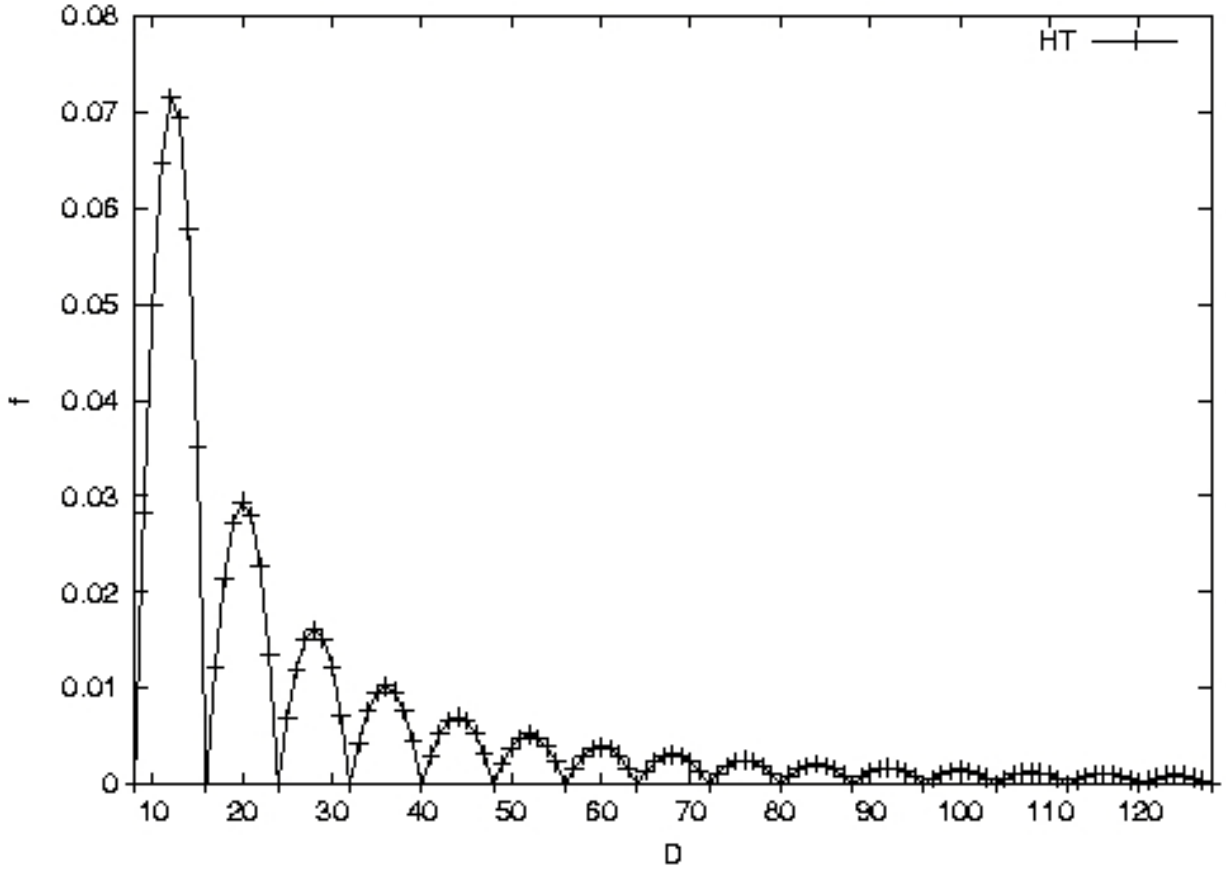
- Unfairness factor in this example for *D=14* is

$$f=5.77\%$$

- A local source has an unfair advantage and gets 5.77% more bandwidth than it should

Unfairness Factor vs. D

- Max. unfairness under *steady state* is 7.14%
- HyperTransport protocol is unfair



Unfairness of *HT* – Observation

- Unfairness of *HT* protocol may be worse if:
  - in consecutive periods, values of *D* vary
  - random number generator is biased

Our *HTfair* Protocol

- Again, consider the example used before
- Rate calculations involve the following counters:
  - $r = D >> 3$ – number of back-to-back remote packets between transmissions of a local packet
    - Example: $r = 14 >> 3 = 1$
  - $c_r = D / gcd(D,8)$ – number of remote packets in a sequence (*gcd* is a greatest common divisor)
    - Example: $c_r = 14 / gcd(14,8) = 14/2 = 7$
  - $c_l = 8 / gcd(D,8)$ – number of local packets in a sequence
    - Example: $c_l = 8 / gcd(14,8) = 8/2 = 4$

HTfair Protocol Operation

- While r *and* c_r are positive, forward remote packets and decrement r and c_r
  - Example: $r = 1 \rightarrow 0$; $c_r = 7 \rightarrow 6$
- If r drops to 0, reset r, send *one* local packet, and decrement c_l
  - Example: $r = 0 \rightarrow 1$, $c_l = 4 \rightarrow 3$
- Repeat it as long as c_r and c_l are positive
- If one of them drops to 0, "finish" the other one forwarding appropriate packets
  - Example: $c_l = 0$, $c_r = 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ – three remote packets sent back to back

HTfair Protocol – Results

- Pattern produced by:
  - *HTfair*: RLRLRLRLRRR
  - *HTideal*: RRLRRLRRLRL
- Transmission rates given by *HTFair* and *HTIdeal* are identical
- Differences:
  - Bursts of remote packets are longer
  - Jitter of local packets is higher
    - Jitter is defined as the standard deviation of inter-departure delays of local or remote packets

HTfair with and without *gcd*

- Assume D=14
- Counters and patterns *with* gcd:
  - $c_r = D / gcd(D,8)=7$, $c_l = 8 / gcd(D,8)=4$
  - RLRLRLRLRRR
- Counters and patterns *without* gcd:
  - $c_r = D = 14$, $c_l = 8$
  - RLRLRLRLRLRLRLRLRRRRRR
- Rates are the same, bursts are longer without gcd

HTfair Implementation

- *HTfair* is similar to *HT* but it:
  - needs two 3-bit counters c_l and c_r
  - needs a reduction scheme by greatest common divisor
  - does not need implementation of random number (linear shift register)
- A lookup table for *gcd* calculation

D	gcd(D, 8)
...000	8
...100	4
...x10	2
...xx1	1

Summary

- We have analyzed fairness of HyperTransport protocol
  - In the steady state, the protocol's unfairness factor may exceed 7%
  - In a variable state, unfairness may further deteriorate
- We proposed *HTfair* algorithm
  - As fair as the ideal algorithm
  - About as simple to implement as *HT*
  - Only slightly increases burstiness
  - Adopted for implementation in a new product

Future Work

- Application of Bresenham's line algorithm to generate patterns of local and remote packets
  - Insertion rate 8/D is in fact a slope of a function that represents a line
  - To generate a pattern, Digital Differential Analyzer (DDA) for linear interpolating could be used
- Simulations

- Wladek Olesinski
- wladek.olesinski@sun.com