Modeling and Optimizing Pipeline Latency

Randy Wang, Arvind Krishnamurthy, Rich Martin

Tom Anderson, David Culler

Computer Science Division, University of California, Berkeley

Department of Computer Science and Engineering, University of Washington, Seattle

October, 1997

Question

What’s the optimal latency of transferring 4K on Myrinet??

State-of-Art

- Many applications care about medium message latency
  - network memory (e.g., GMS)
  - distributed file system (e.g., xFS)
  - database applications
- Lack of research in this area
  - focus:
    e.g., previous emphasis on small and bulk messages
  - problem domain:
    e.g., different problem(s) on the internet
  - approach:
    e.g., GMS study used brute force simulation + ad hoc hacks for a particular configuration

Problem Statement

- want to send a ***packet***
- can break it down into ***fragments***
- the network has a number of ***pipeline stages***
- each stage is defined by:
  - a ***per-fragment startup cost***, and
  - an additional ***per-byte cost***
- each stage is ***store-and-forward***
- question: how to break down the packet to minimize ***latency*** of the entire packet?

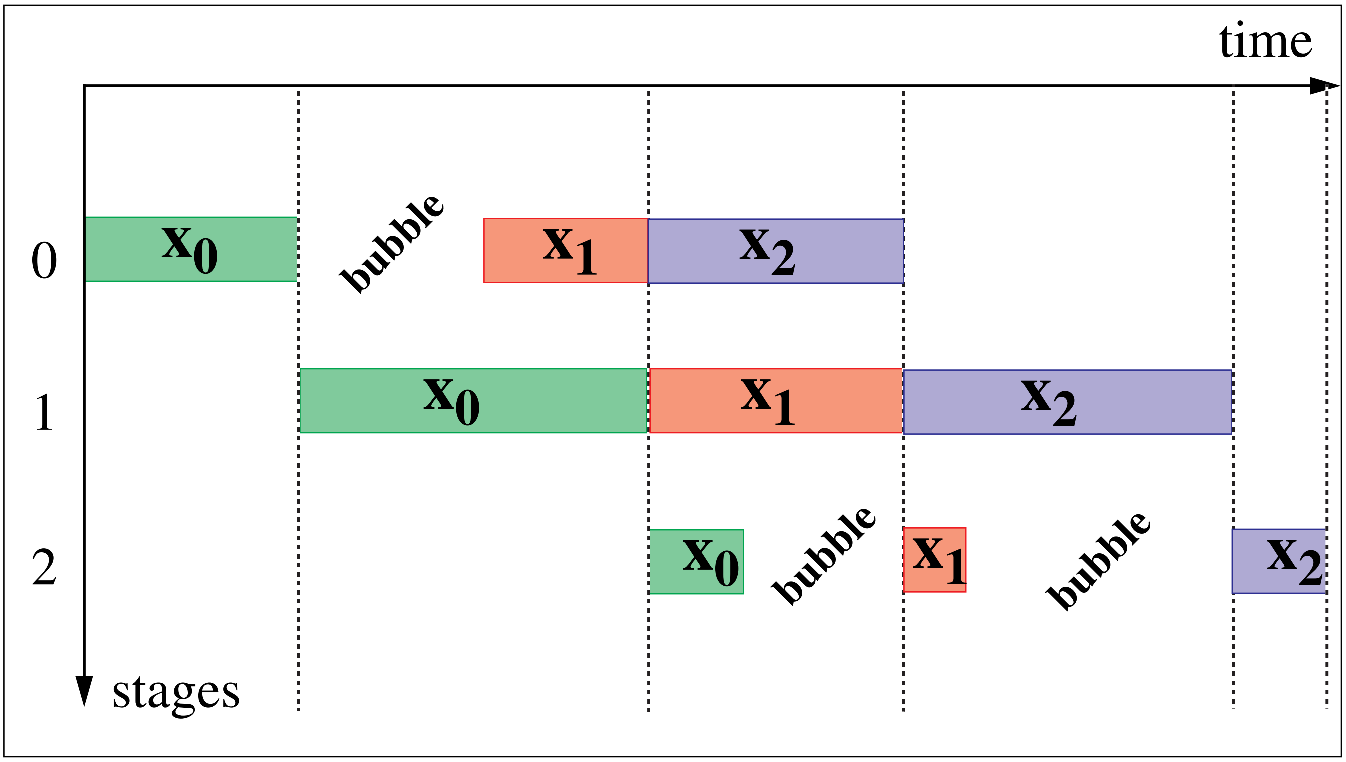
Outline

- Introduction
- ***Algorithms and models***
- Experimental methodology and results
- Conclusions

Algorithms

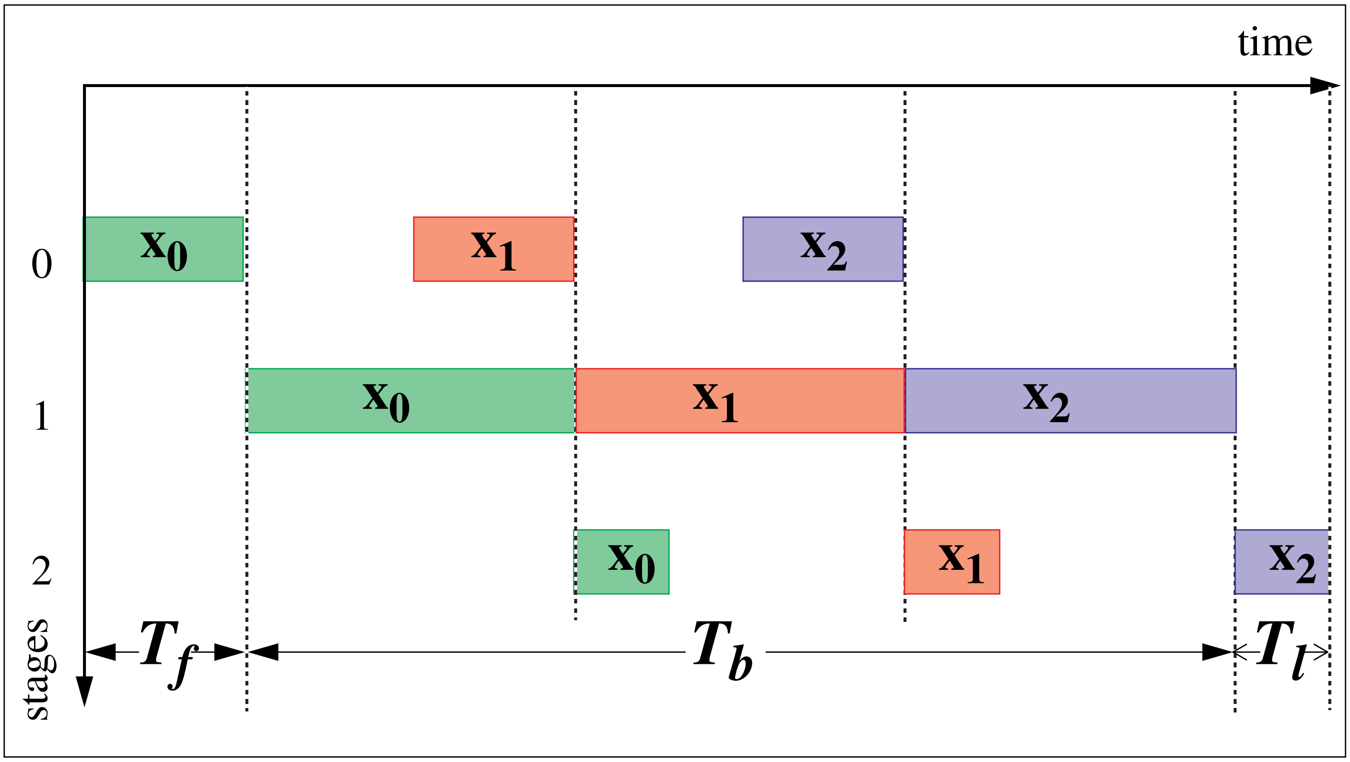
- Linear (integer) programming
- Fixed-sized fragmentation
- Variable-sized fragmentation
  - 2-stage pipelines
    no-stall, ramp-up, reversibility theorems
  - 3-stage fast-slow-fast pipeline
    ramp-up-down theorem
  - 3-stage fast-slow-slowest pipeline
    hierarchical fragmentation
  - n-stage pipelines
    put it all together!

Notations



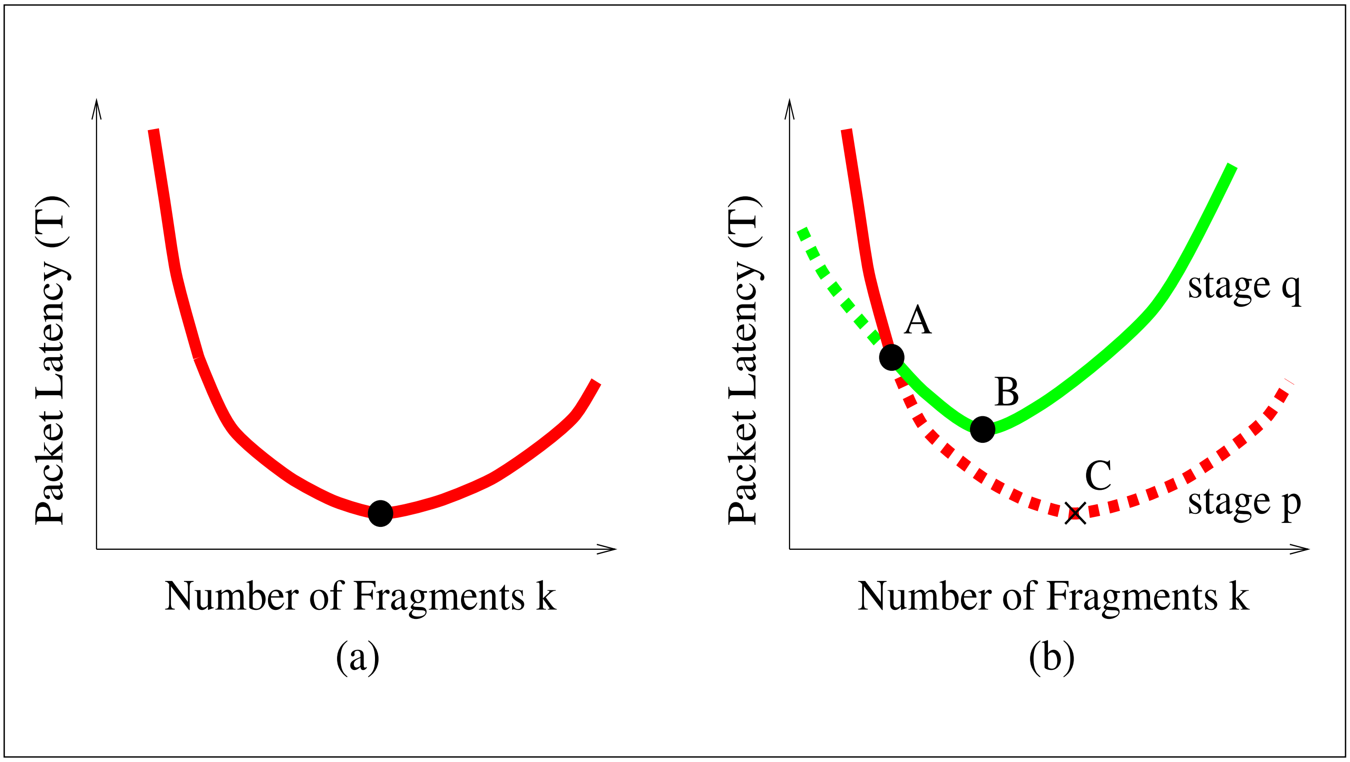
- packet = B bytes
- fragments: $x_0+x_1+...+x_k = B$
- stage i is defined by (g_i, G_i) = (per-fragment overhead, per byte cost)
- bottleneck stage is (g_b, G_b)

Fixed-sized Fragments



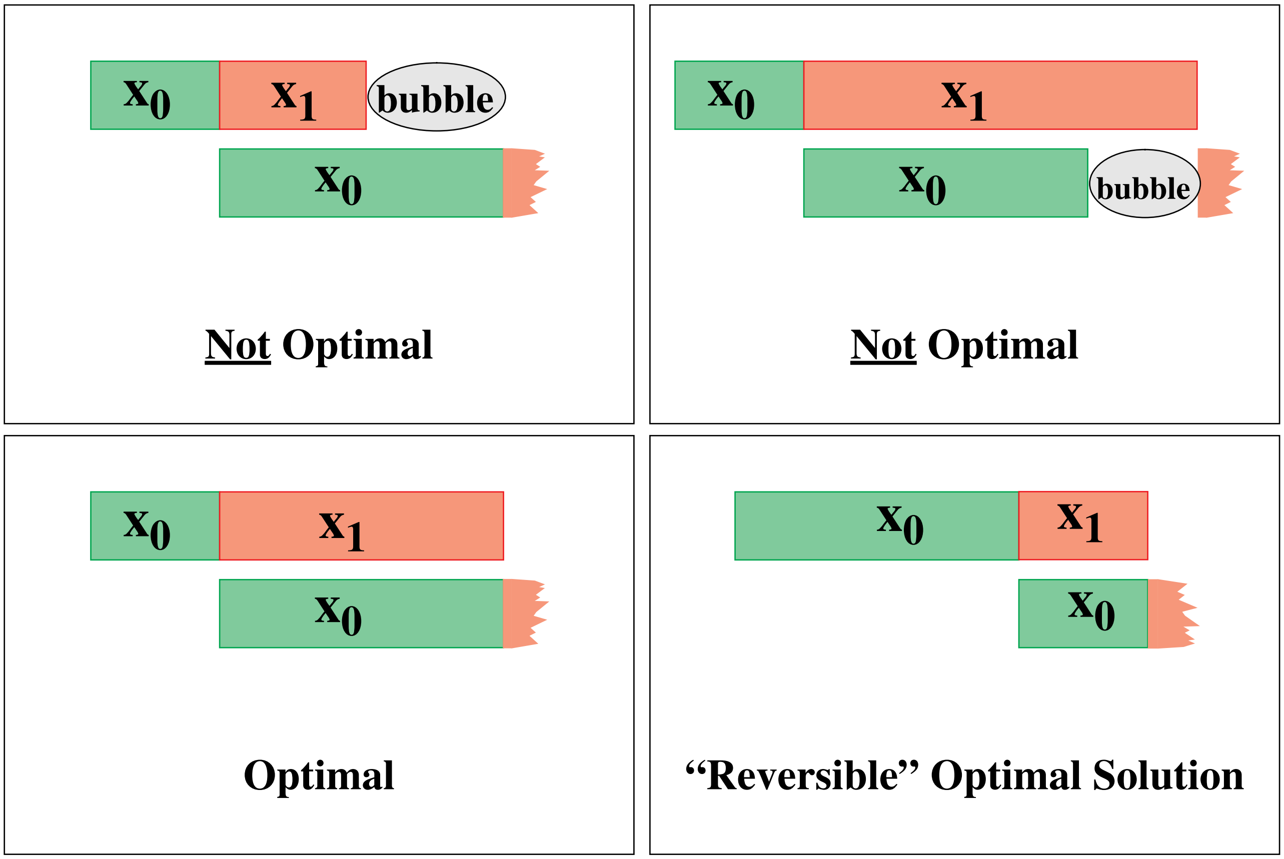
- Must keep bottleneck busy
- ***Key trade-off*** between $(T_f + T_l)$ and T_b
- Total latency is a function of x_i
- Optimal fragment size: $x_i = \sqrt{\dfrac{B \bullet g_b}{\sum_{i \neq b} G_i}}$

Generalizing Bottleneck Definition



- ***Signature curve*** of a bottleneck
- ***Latency curve*** of a pipeline = concatenation of multiple signature curves
- Find ***global minimum*** on latency curve

2-stage Pipelines

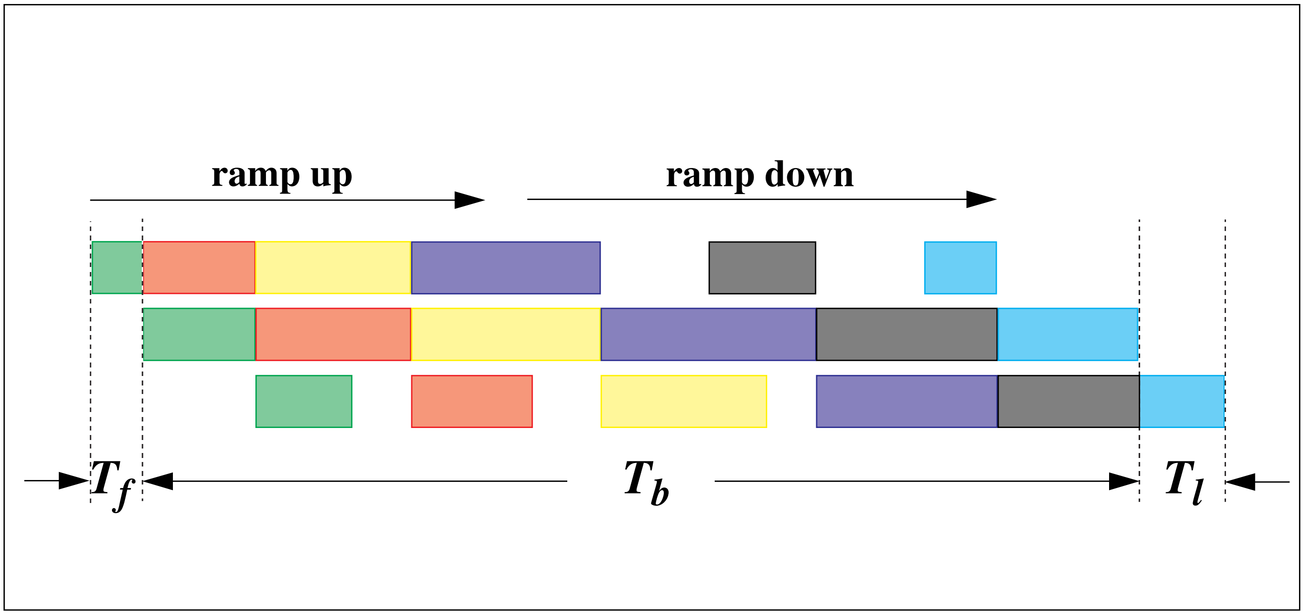


- Optimal ***only if*** no bubble (Ramp-up Theorem):

$$x_{i+1} = \frac{g_1 - g_0}{G_0} + \frac{G_1}{G_0} \bullet x_i$$

- Latency is function of x_0 (find best x_0)
- "Reversible" optimal solutions for "reversed" pipelines ("Reversibility Theory")

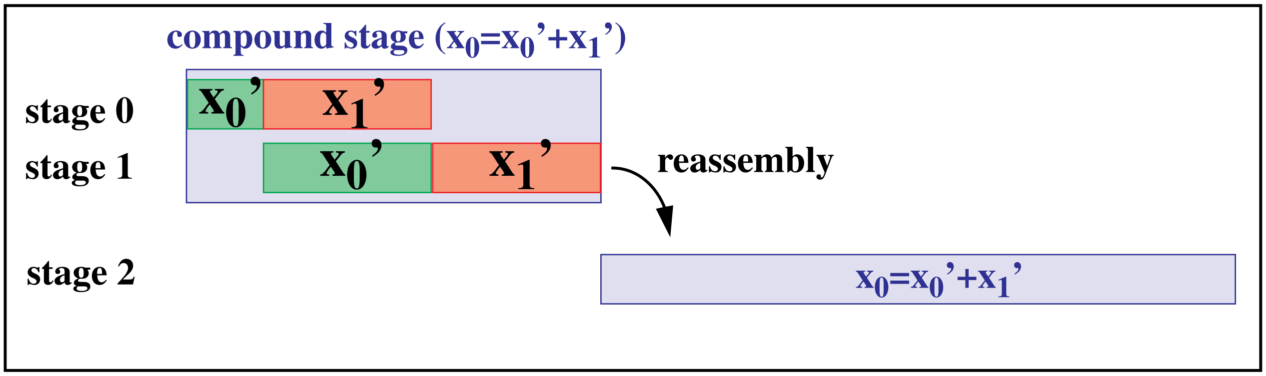
3-stage Pipeline
(fast-slow-fast)



Intuition of ramp-up-down:

- $T = T_f + T_b + T_l$
- Want small T_f -- start with small frag., and "ramp up"
- Want small T_l -- end with small frag., so "ramp down"

3-stage Pipeline
(fast-slow-slowest)



- Case for (possible) reassembly
- Hierarchical fragmentation
  - treat the first two stages as a ***compound*** stage
  - search for optimal x_0 for overall pipeline
  - apply recursion to breakdown x_0 for sub-pipeline
- Reversibility Theory solves slowest-slow-fast case (fragmentation instead of reassembly)

Algorithm Summary

- Fixed-sized fragmentation
  - good for pipeline whose stages are similar
- Ramp-up, Ramp-up-down fragmentation
  - optimal for 2-stage and fast-slow-fast pipelines
  - provides upper bound and insight
- Hierarchical fragmentation
  - useful when one stage is much slower than others

General Pipeline
(Put it all Together)

- Break pipeline into three parts:
  bottleneck stage,
  the stages before, and
  the stages after
- Treat the before and after stages as
  two compound stages
- Apply recursion to each of the compound stages
- Apply fixed-sized fragmentation,
  or ramp-up-and-down fragmentation

Outline

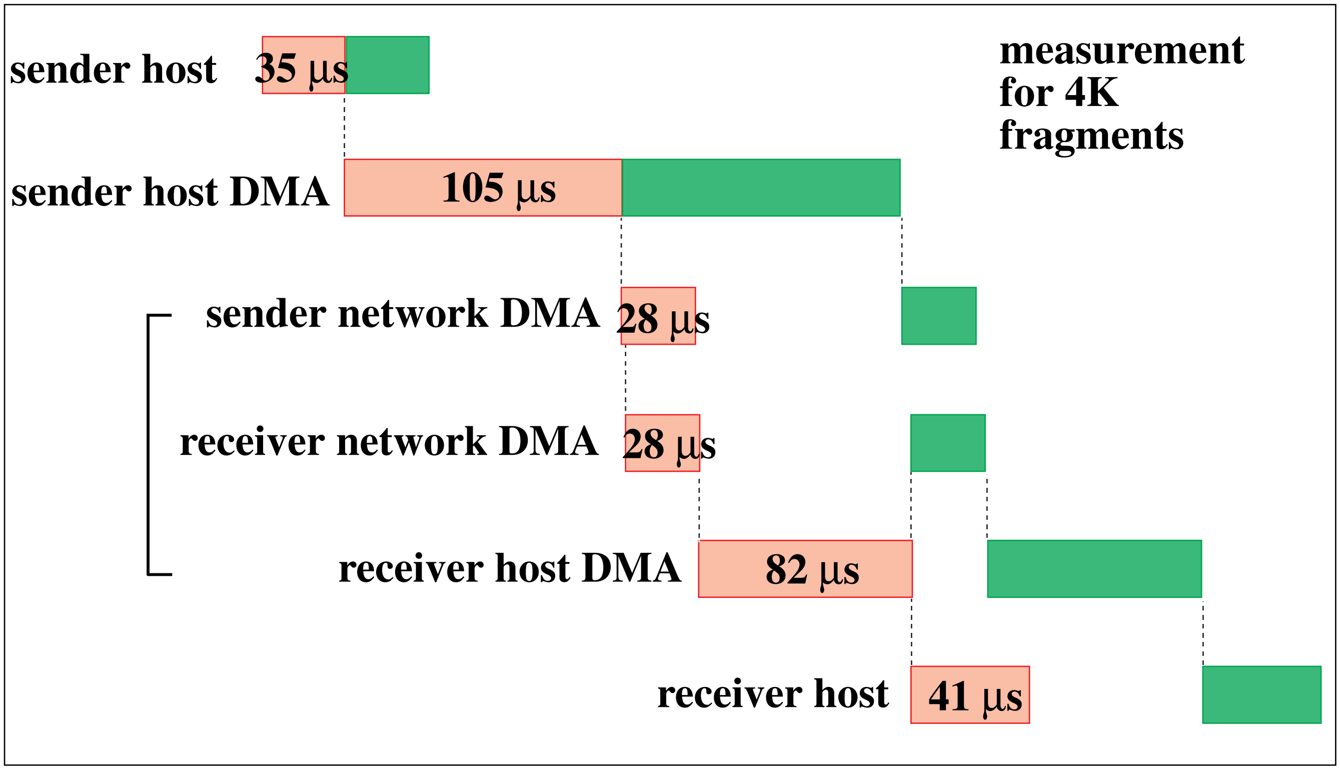
- Introduction
- Algorithms
- ***Experimental methodology and results***
  - methodology
  - clear-box and black-box measurement of pipeline parameters
  - evaluation of fixed-sized fragmentation
  - evaluation of variable-sized fragmentation
- Conclusions

Methodology

- Obtain pipeline parameters
  - clear-box instrumentation
  - black-box measurement
- Construct customized pipeline algorithms
- Adapt to user packet sizes and changing network characteristics at run time

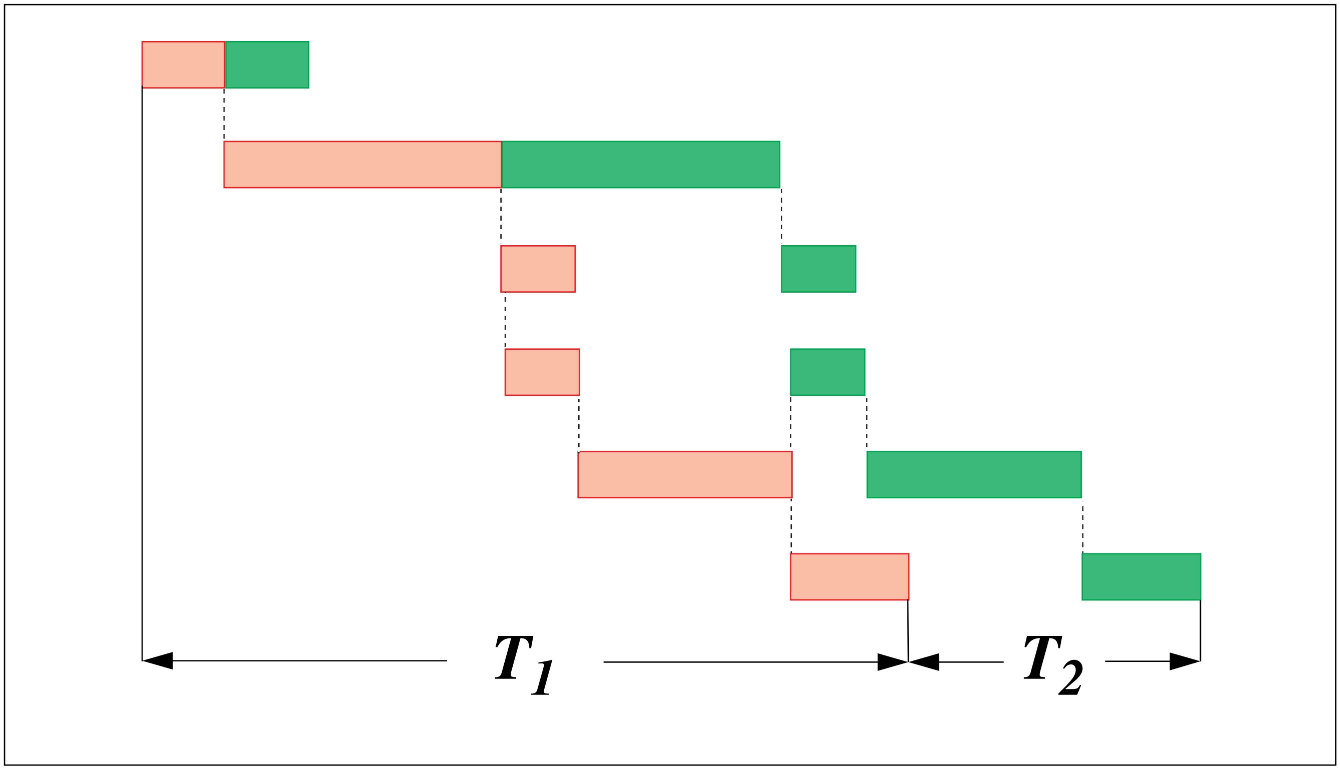
Myrinet-GAM Pipeline

(Clear-box Instrumentation)



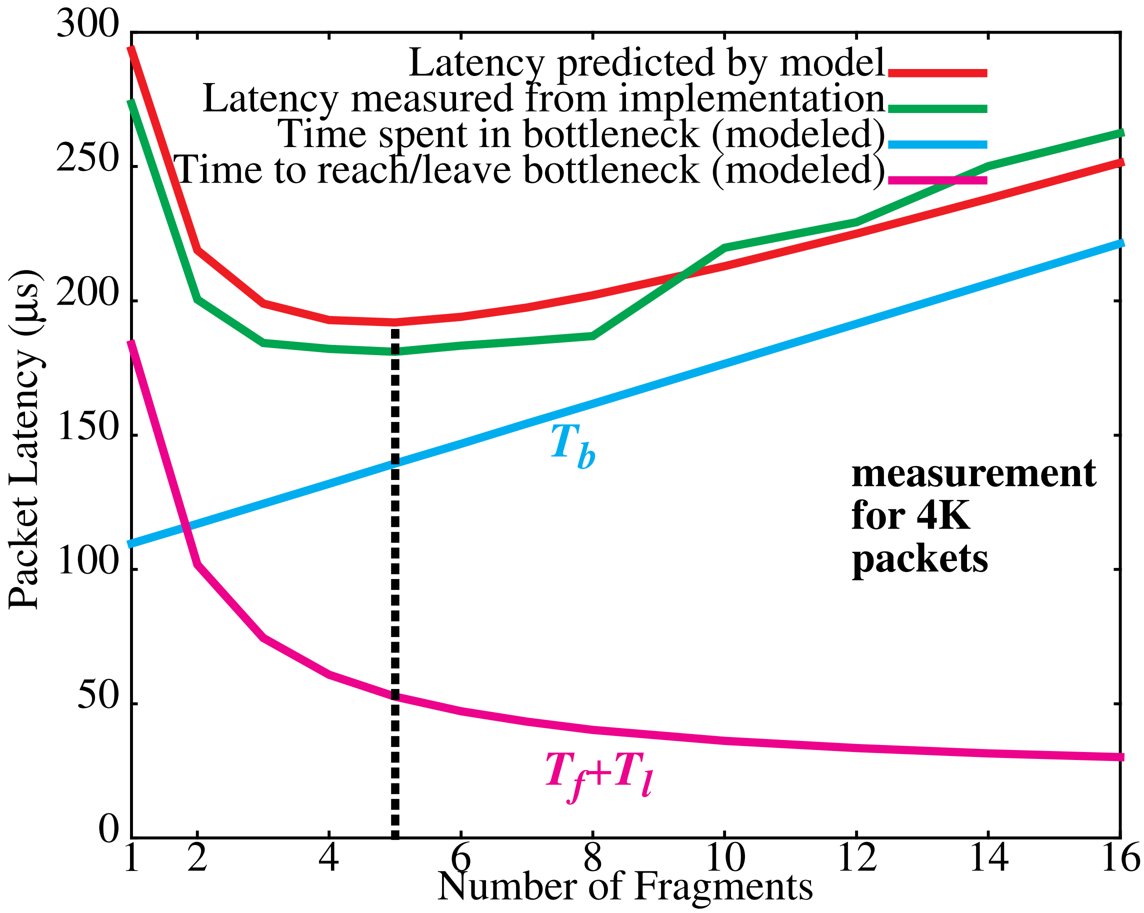
stage i	g_i (µs)	G_i (µs/kB)
0	7.2	7.2
1	5.2	24.9
2	7.5	24.9
3	7.4	7.9

Myrinet-GAM Pipeline
(Black-box Measurement)



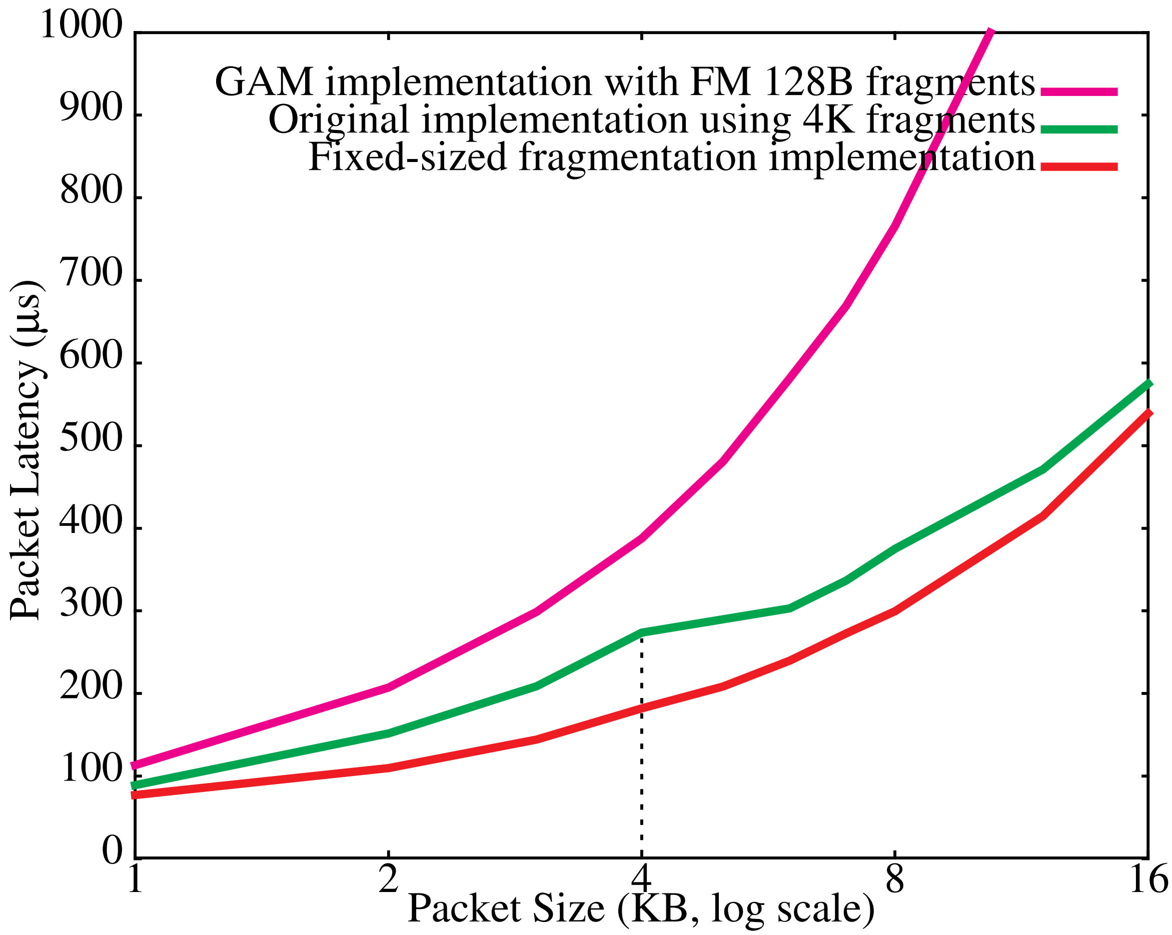
- Measure $T_1 = (\Sigma g_i + x \cdot \Sigma G_i)$ to obtain $(\Sigma g_i, \Sigma G_i)$
- Measure $T_2 = (g_b + x \cdot G_b)$ to obtain (g_b, G_b)
- $(g_b, \Sigma G_i - G_b)$ sufficient for fixed-sized fragmentation

Validation of Fixed-sized Model



- Model error averages 5.9%
- Tradeoff between (T_f+T_l) and T_b

Applying Fixed-sized Fragmentation

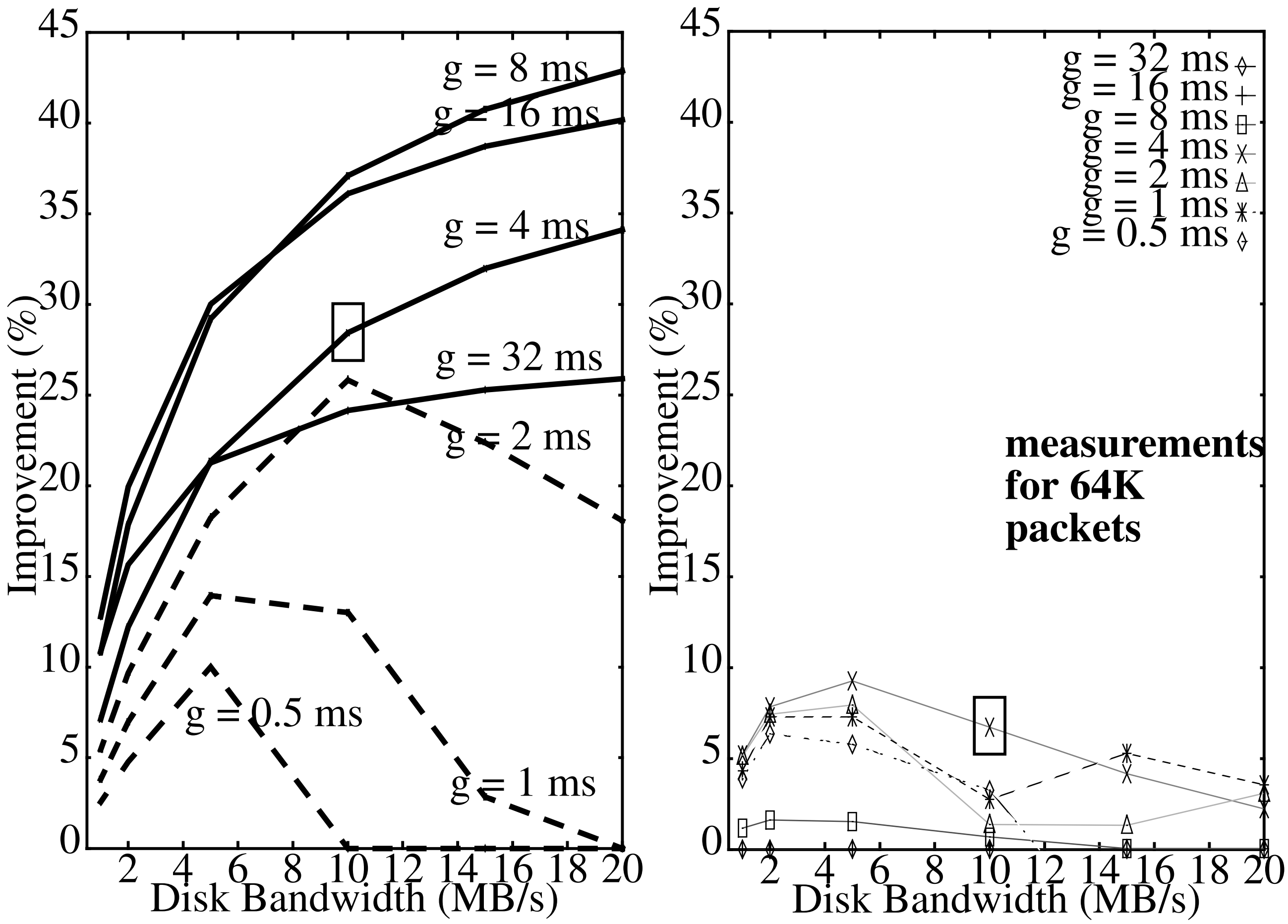


- Adaptive fragment size by fixed-sized fragmentation
- for 4KB, 51% improvement over original implementation

Evaluation of Variable-sized Fragmentation

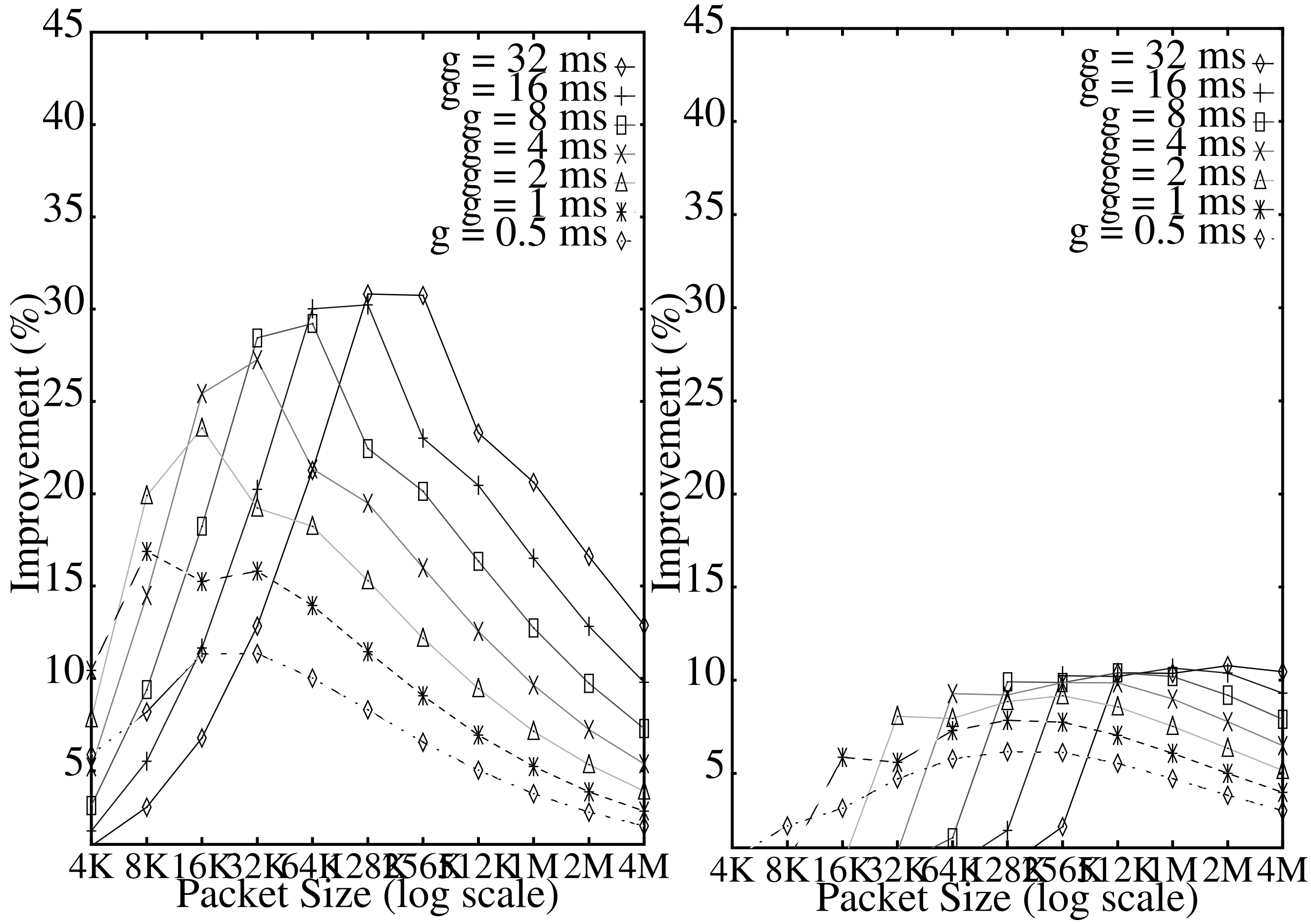
- Two pipelines
  - network - disk
  - network - disk - network
- Three algorithms
  - A: simple fixed-sized fragmentation
  - B: add hierarchical fragmentation
  - C: add ramp-up fragmentation
  - C’: add ramp-up-down fragmentation

Sensitivity of Variable-sized Fragmentation



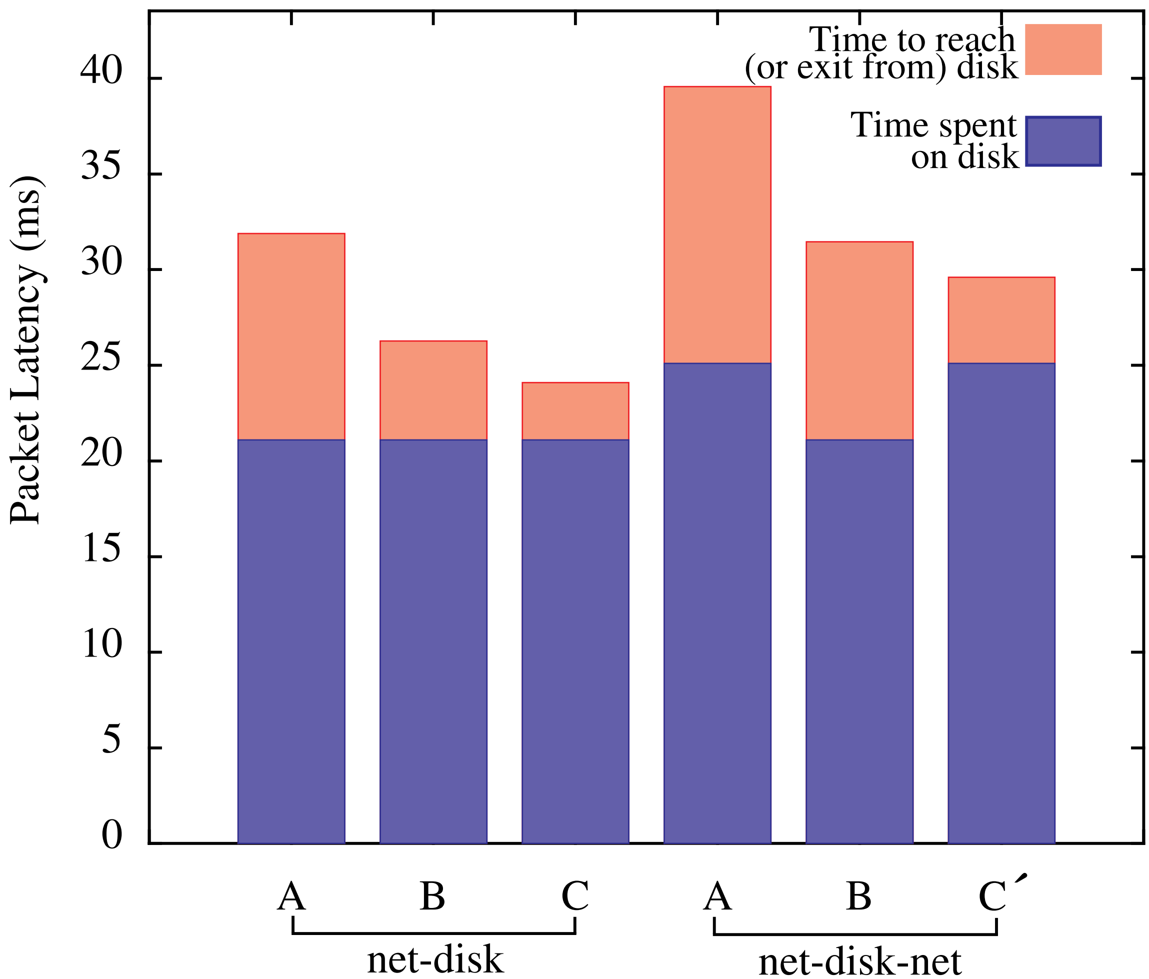
- Hierarchical fragmentation good when bottleneck
  - g ~ an order of magnitude worse
  - G ~ no worse than an order of magnitude larger
- Ramp-up fragmentation: < 10% improvement

Sensitivity of Variable-sized Fragmentation to Packet Size



- Hierarchical fragmentation good for medium messages
- Ramp-up fragmentation: < 11% improvement

Timing Breakdown of Variable-sized Fragmentation



- Main effect is reducing (T_f+T_l)
- Fixed-sized + hierarchical fragmentation is a good approximation

Conclusion

- Medium messages are important
- Series of pipelining algorithms based on a single tradeoff between (T_f+T_l) and T_b
- No more
  - "voodoo constants"
  - brute force simulation
- Instead, the algorithm dynamically adapts to
  - user packet size, and
  - network characteristics
- Fixed-sized + hierarchical fragmentation is a good approximation