A Content Fetching Layer over NDN (with demo)

Jun Bi

Xiaoke Jiang, Zhaogeng Li, Ao Guo

Tsinghua University/CERNET

Motivation

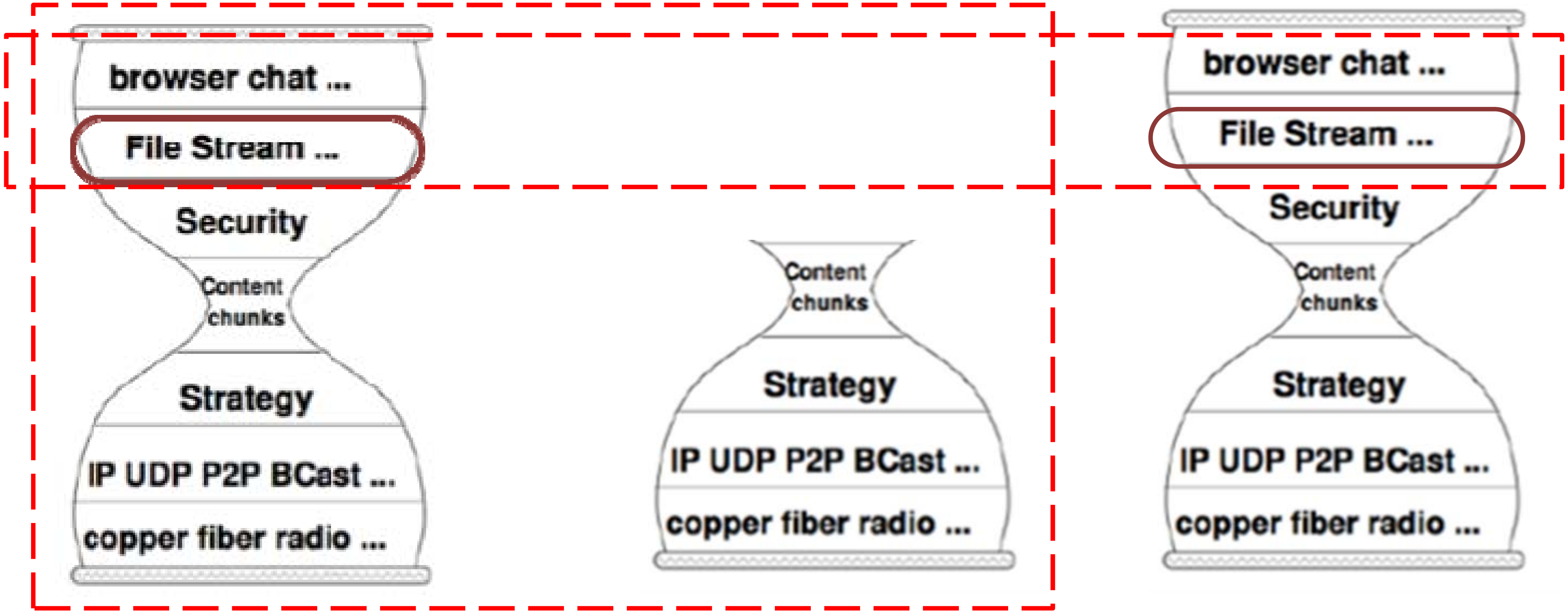
- In NDN application layer, applications need to get a sequence of chunks of “application content” (content concept at application layer)
  - E.g, File: error-free
  - Video streaming: real-time
- So we need a layer over NDN, to serve for applications to fetch “application content” with better performance
  - We call it *Content Fetch Layer*

Motivation

NDN End System
(e.g. content consumer)

Network
NDN Router

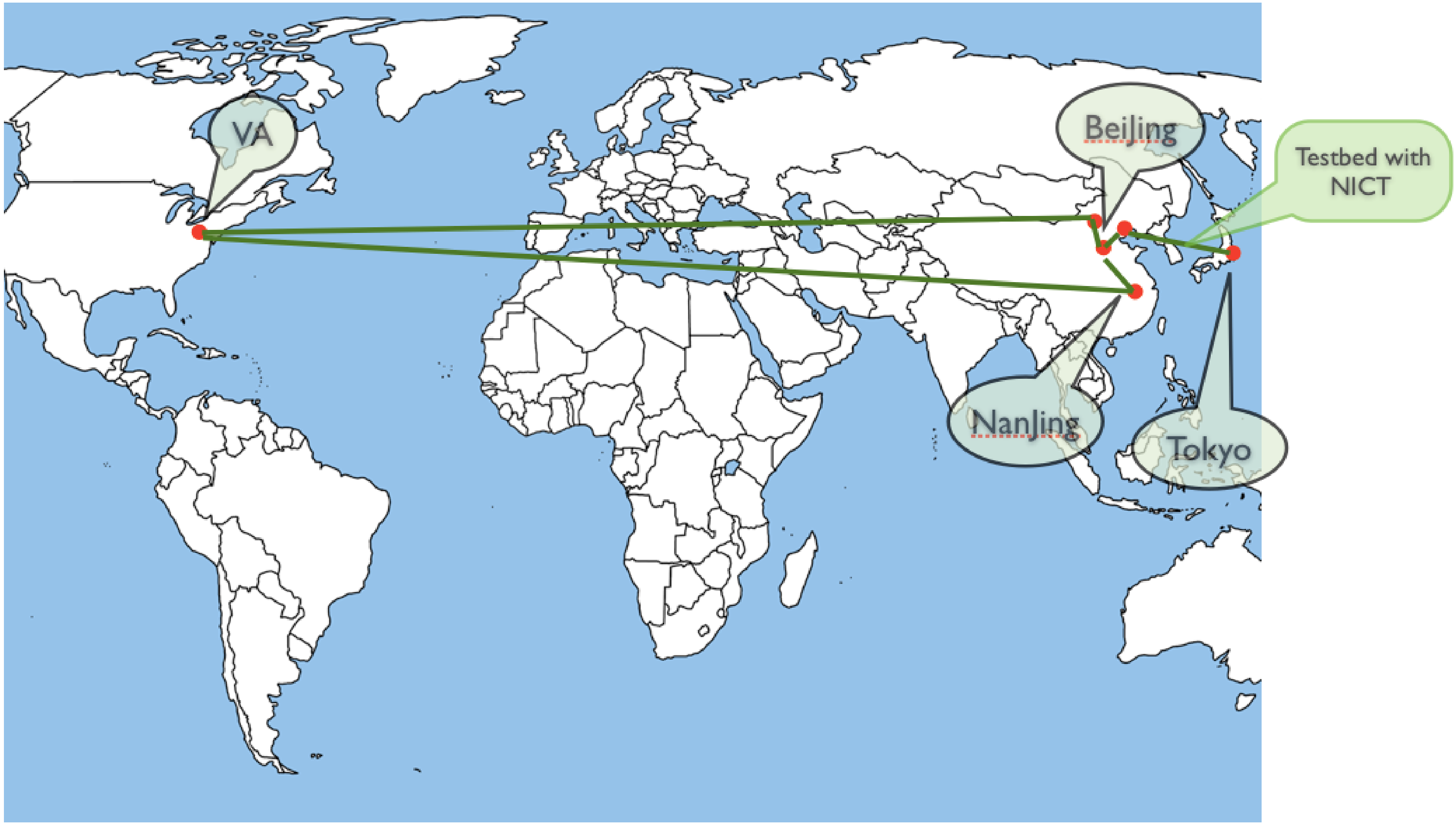
NDN End System
(e.g. content provider)



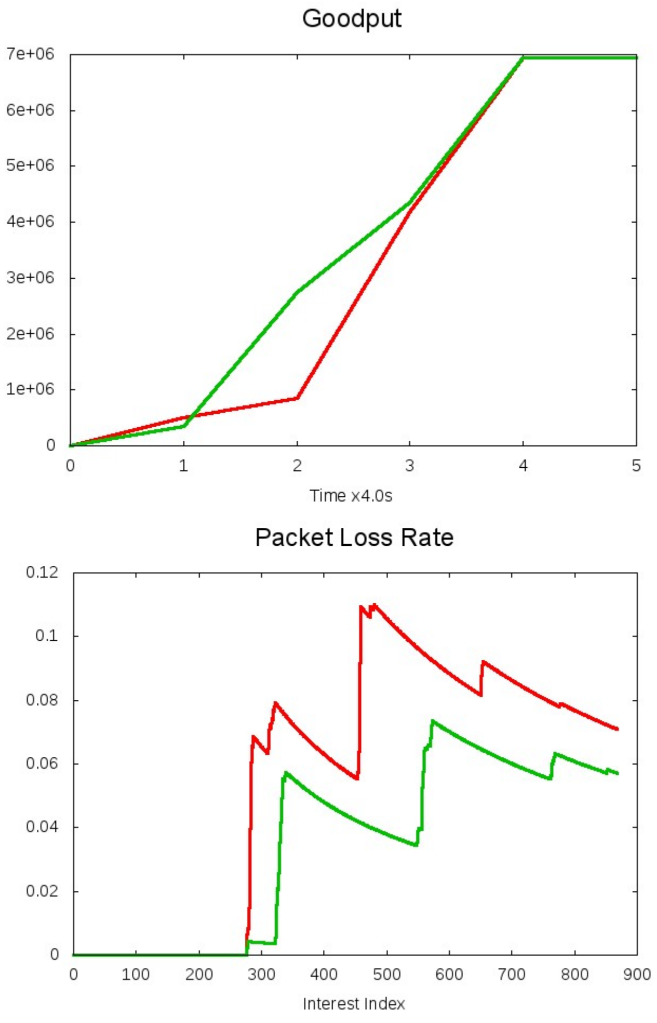
Content Fetching Layer

- To improve performance for retrieving a sequence of chunks for the same “application content”
  - consumer-to-network mode (for most applications)
    - Adaptive Congestion Window
      - Slow Start, AIMD; or new mechanisms
    - Adaptive Interests’ lifetime (RTO: Retransmission Timeout)
      - RTO = α*SRTT+β*RTTVar
      - α makes NDN routers have more time to try alternative paths for interests, β makes the RTO adaptive to network condition (faster re-retransmission request)
  - end-to-end mode (only for point-point applications) and (might) provider-to-network mode (e..g online video provider)
    - Adaptive Chunk Size
      - optimized by keep tracking chunk loss rate

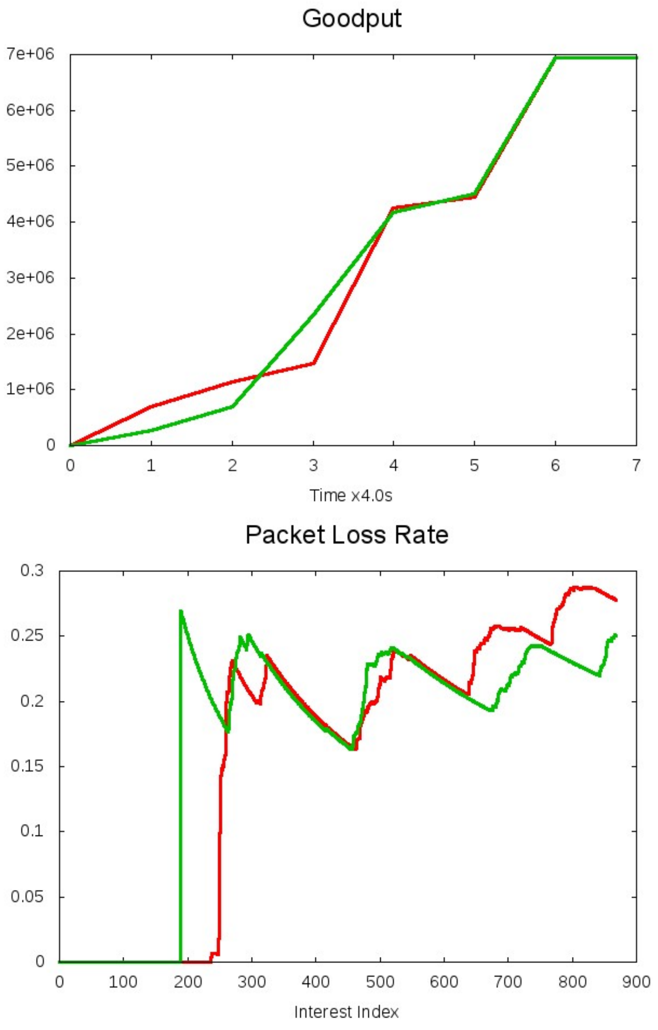
Experiment Topology (Implemented with NDNx)



NDN without “content fetch layer”



WIN=256



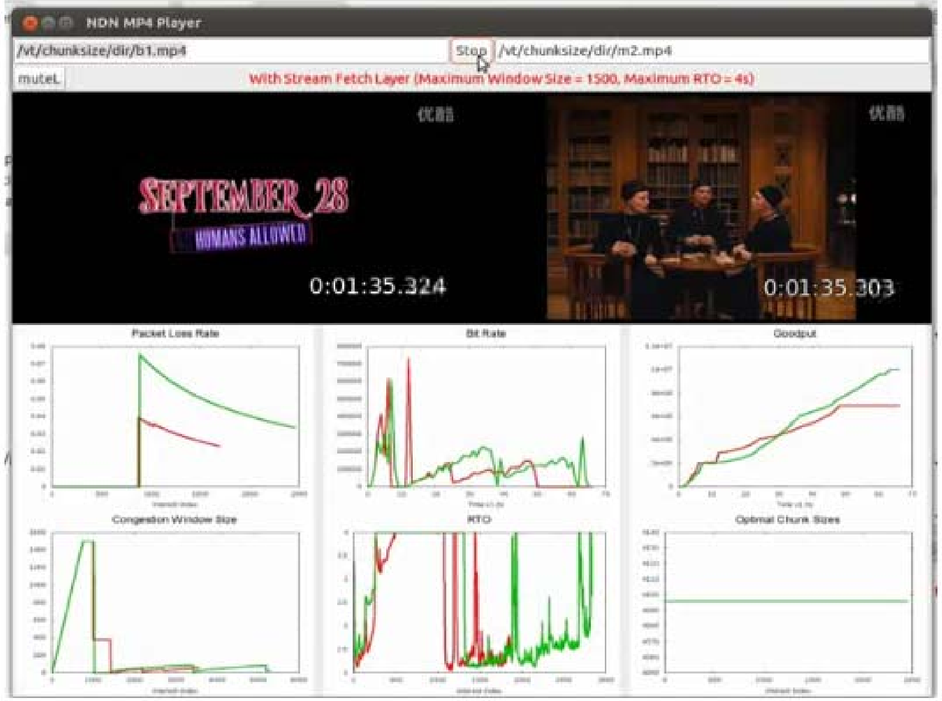
WIN=512

High
Loss
Rate

Demo: Adaptive Window-size/RTO

Max WindowSize=1500
Max RTO=4s
ChunkSize=4KB

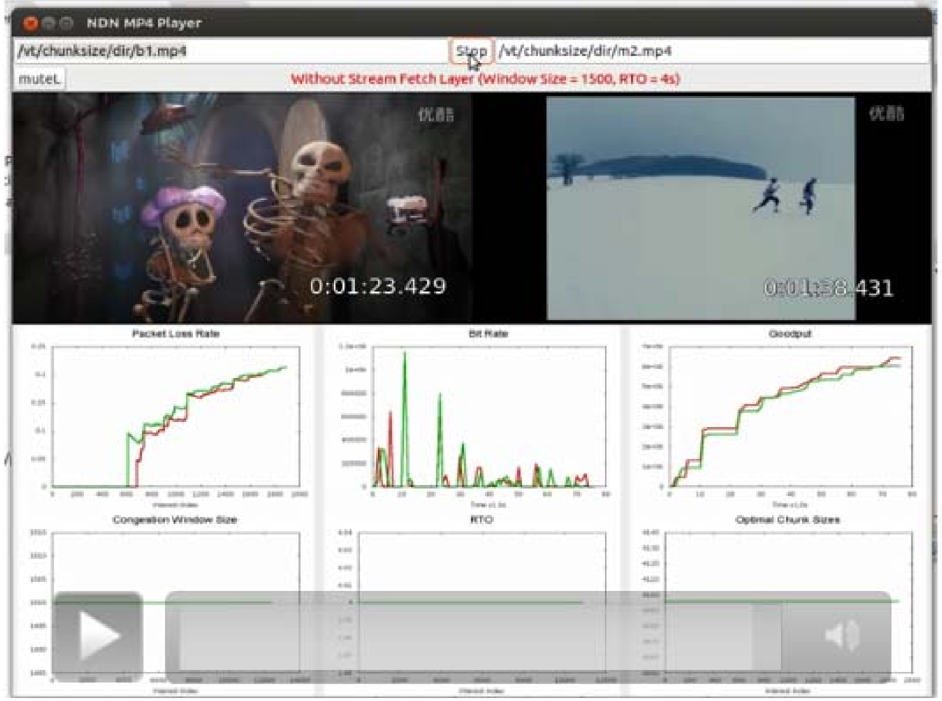
Fixed WindowSize=1500
Fixed RTO=4s
ChunkSize=4KB



/vt/chunksize/dir/b1.mp4

/vt/chunksize/dir/m2.mp4

With Stream Fetch Layer (Maximum Window Size = 1500, Maximum RTO = 4s)



/vt/chunksize/dir/b1.mp4

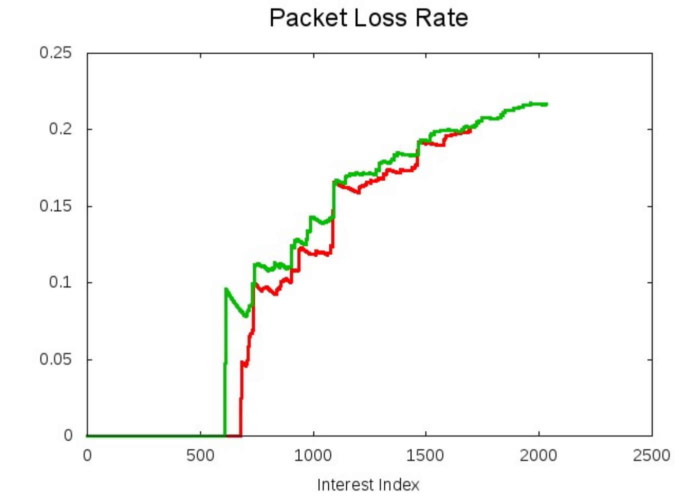
/vt/chunksize/dir/m2.mp4

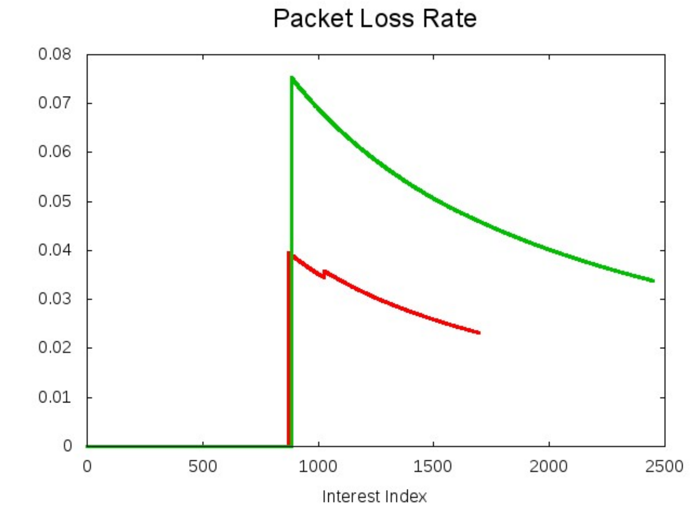
Without Stream Fetch Layer (Window Size = 1500, RTO = 4s)

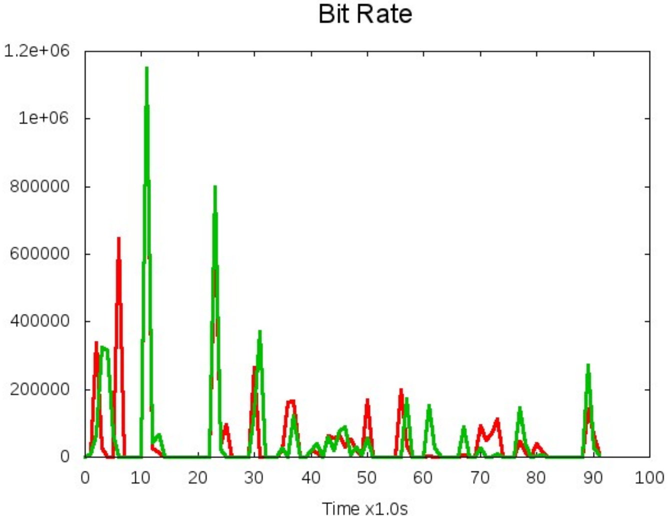
Loss Rate Bit Rate Goodput

Window Size RTO Chunk Size

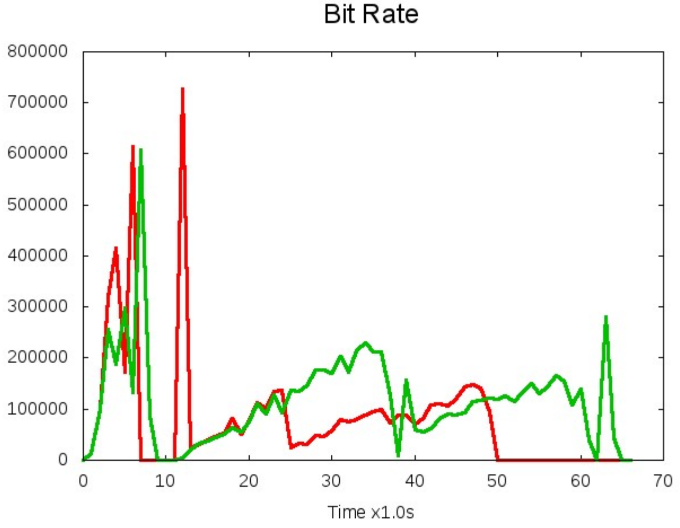
Comparison 1





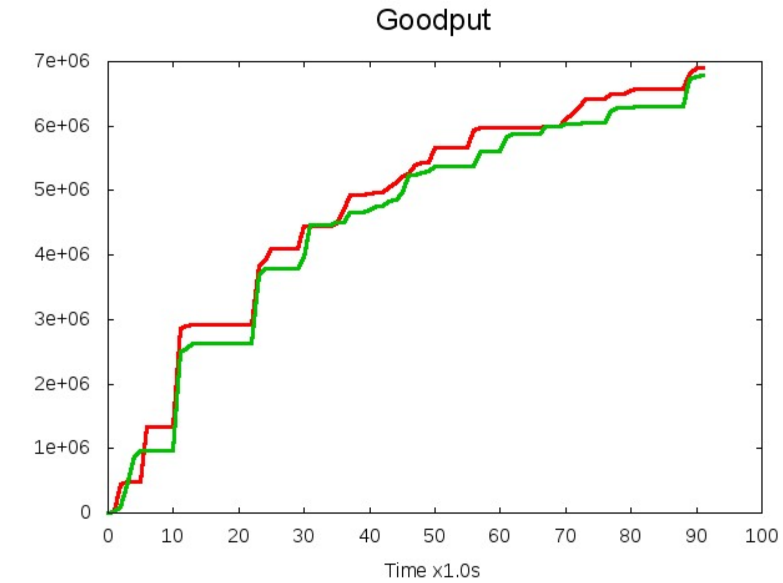


Fixed



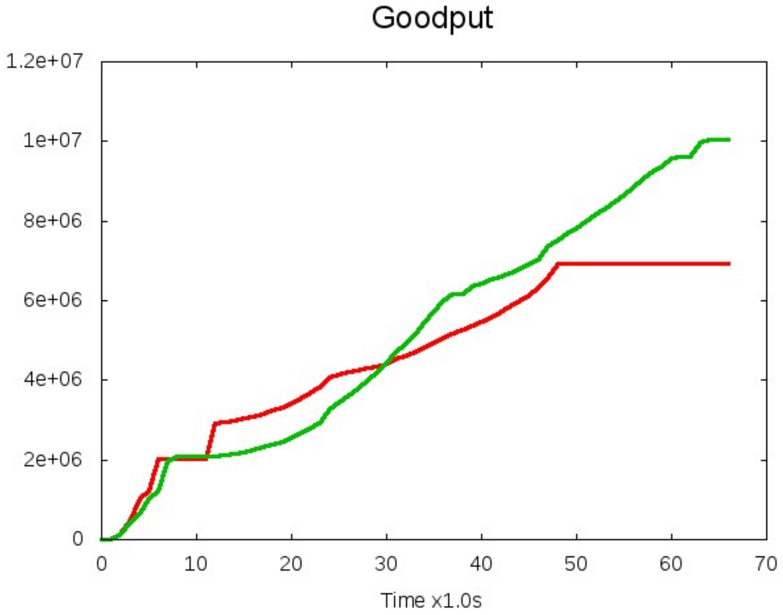
Adaptive

Comparison 2



Fixed WindowSize=1500
Fixed RTO=4s
ChunkSize=4KB

Adaptive window size and RTO will improve the Goodput.

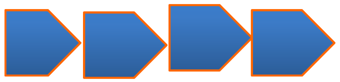


Max WindowSize =1500
Max RTO=4s
ChunkSize=4KB

Adaptive Chunk Size Mechanism

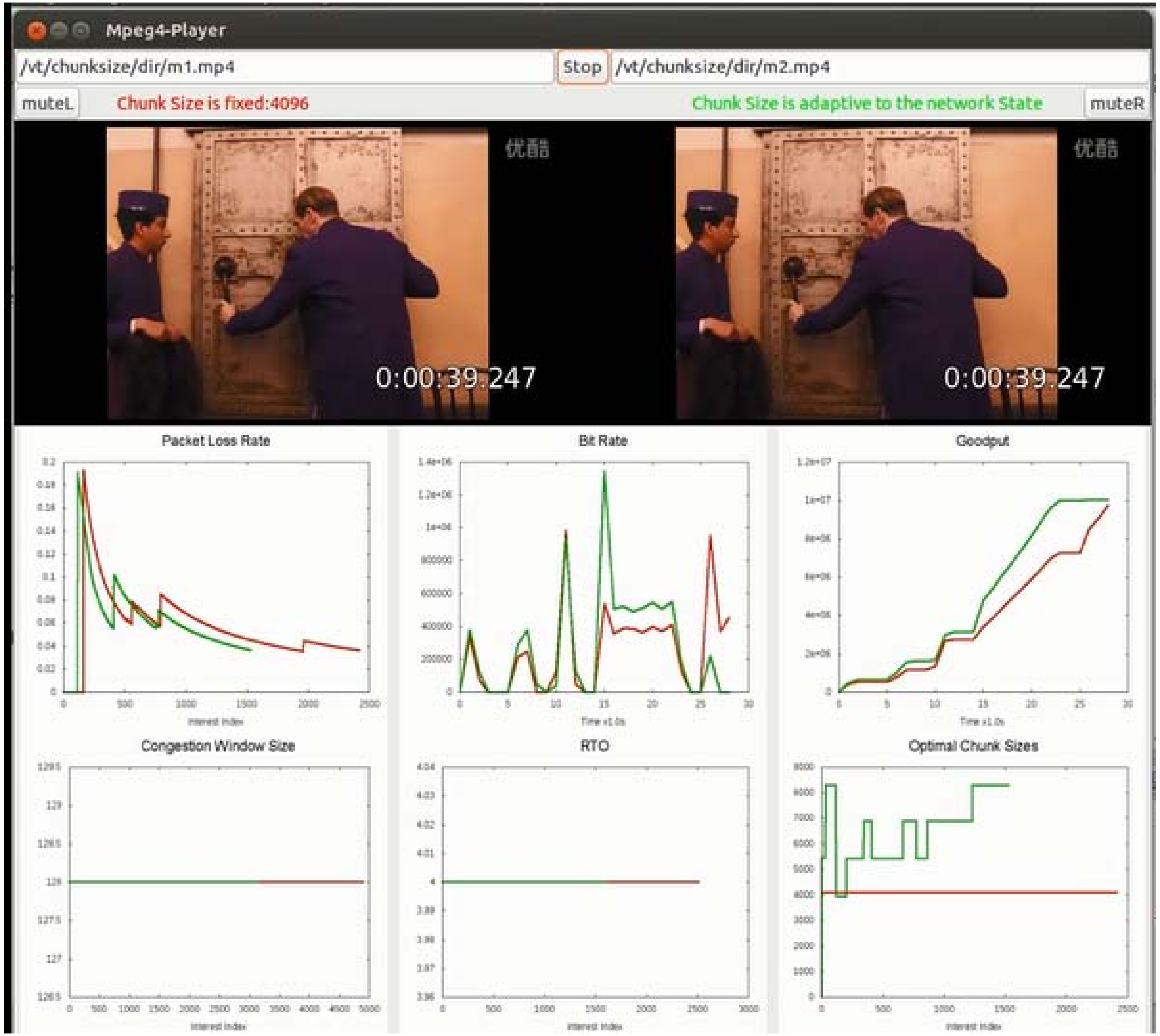
- Chunk Size
  - too small: low payload ratio
    - large header overhead, ~500B: signature, key locator, ...
  - too big: high chunk loss rate and more re-transmission
    - underlying layers need to do fragmentation
    - one piece of fragmentation loss will cause the whole big chunk loss

Why Adaptive Chunk Size

- Say, the chunk header size is 1, underlying packet size is 3; and size of requested "application content" is 8. underlying packet loss rate p
  - Solution1: 4 chunks:  chunk size=2+1
  - Solution2: 1 chunks: chunk size=8+1

	Solution 1	Solution 2
Payload Ratio	2/3=66.7%	8/9=88.9% ✔
Chunk Loss Rate (p=1%)	1% per chunk ✔	1-(1-p)^3 = 2.9%
Chunk Loss Rate (p=3%)	3% ✔	8.7% ✗
Size if Retransmission	3	9 ✗

Demo: Adaptive Chunk Size Mechanism



/vt/chunksize/dir/m1.mp4

/vt/chunksize/dir/m2.mp4

Chunk Size is fixed:4096

Chunk Size is adaptive to the network State

Summary

- We may need a “content fetch layer” at end systems to better serve applications
- Some possible mechanisms in this content fetch layer were implemented in our prototype and testbed
  - end-to-network mode (for most applications)
    - Adaptive Congestion Window
    - Adaptive Interests’ lifetime (RTO)
  - end-to-end mode (Only for point-point applications) and (might) provider-to-network mode (e..g online video provider)
    - Adaptive Chunk Size
  - Maybe more functions

Thanks !