Increase-Decrease Congestion Control for Real-time Streaming: Scalability

Dmitri Loguinov
City University of New York

Hayder Radha
Michigan State University

Motivation

- Current Internet video streaming is often constant bitrate (CBR), where the rate is determined a-priori
- A user has to click on the desired rate on a web page
- Typical bitrate selections are quite coarse (i.e., 28k, 56k, 100k, 300k)
- Rate adaptation used in current streaming:
  - Drop one layer upon congestion
  - Add one layer to probe for new bandwidth
- Need protocols that can scale video to any bitrate (such as 435 kb/s) while maintaining fairness properties of regular congestion control

Motivation (cont'd)

- Given MPEG4 FGS (Fine Granular Scalability), can we apply true congestion control to it?
- MPEG4 FGS:
  - One low-bitrate base layer
  - One high-bitrate enhancement layer
  - Rescale the enhancement layer by discarding a certain percentage of each frame
  - Rate-based streaming
- Can rate-based congestion control scale to a large number of concurrent flows?
- "Scalability" applies to the number of flows only

Overview of the Talk

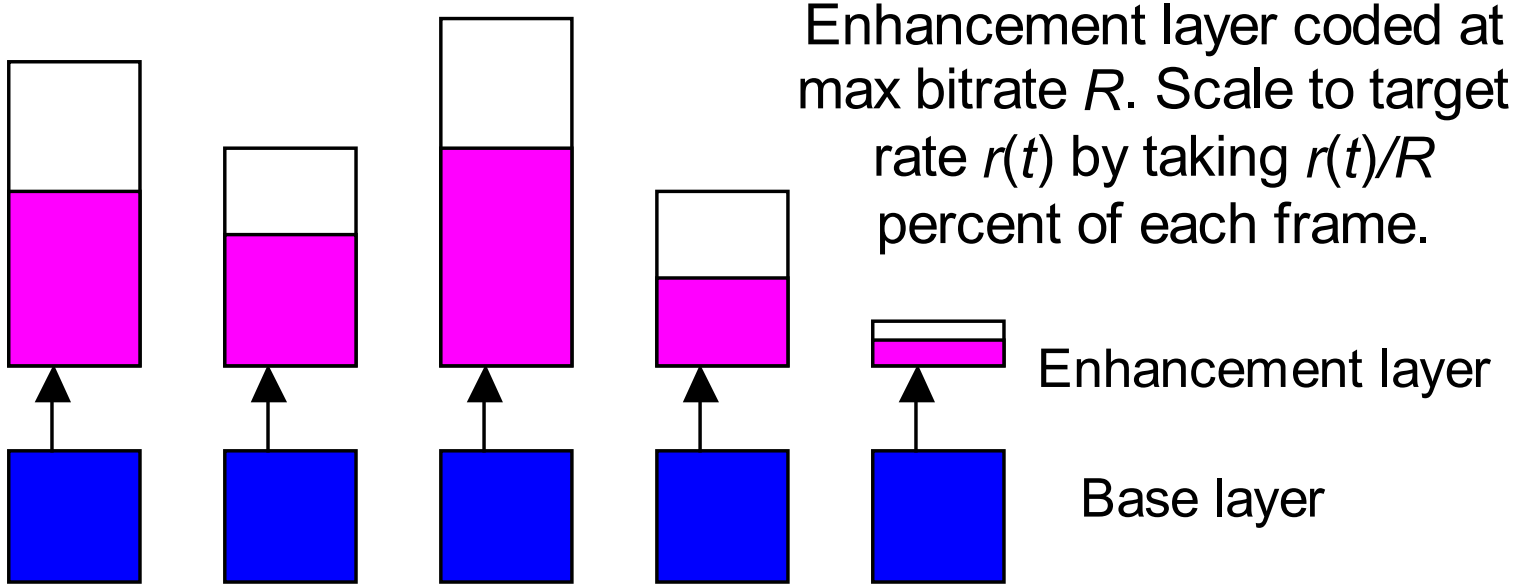
- Background on congestion & flow control
- Scalability of congestion control
- Study of existing methods in NACK-based protocols
- Limitations of the existing methods
- Ideally-scalable congestion control
- Simulations and experimental results
- Drawbacks
- Conclusion

Background (Overview)

- *Flow control* is how application sends its packets
- Two types of flow control
- ACK-based (window-based):
  - receiver acknowledges each *received* packet
  - new packets are sent *only* if ACKs are being received
  - average sending rate $r(t)$ arbitrarily fluctuates
  - sender does not know its average rate $r(t)$ a-priori
- NACK-based (rate-based):
  - receiver acknowledges each *lost* packet
  - new packets are sent regardless of the NACKs
  - average sending rate $r(t)$ is known in advance

Background (cont'd)

- ACK-based flow control is typically difficult to use in real-time streaming:
  - Server must maintain a certain streaming rate for the base layer
  - Difficult to decide how to scale the enhancement-layer pictures since *future* rate $r(t)$ is not known

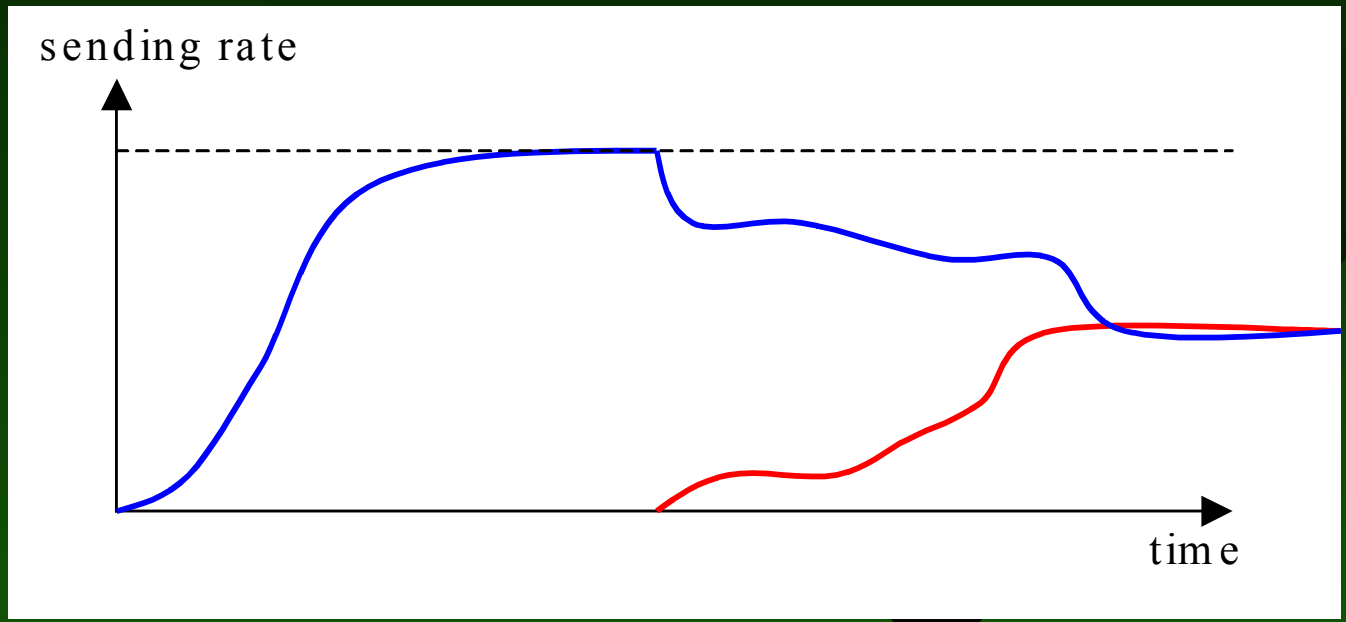


Background (summary)

- We look at the problem of providing end-to-end congestion control for rate-based applications
  - NACK-based congestion control is usually labeled as simply “difficult”
  - ATM/data-link methods exist, but they are inapplicable on the IP+ layers
- We assume a best-effort Internet environment
- QoS (Quality of Service) router support is not available

Congestion Control (overview)

- Typically many flows share common links of *finite* capacity
  - After observing packet loss, flows must reduce their rates
- A key aspect of congestion control is *convergence to fairness*:



Overview (cont'd)

- Internet protocols use congestion control that:
  - reacts to packet loss by *decreasing* the rate
  - reacts to the absence of packet loss by *increasing* the rate
- Congestion control is executed on discrete timescales of RTT (round-trip delay) units each
- Sending rate during interval i is given by r_i

Overview (cont'd)

- Increase-decrease congestion control is summarized as following:

$$r_{i+1} = \begin{cases} r_i + R_I(r_i), \ no \ \ loss \\ r_i - R_D(r_i), \ loss \end{cases}$$

- R_I is the increase function, R_D is the decrease function
- Not all functions guarantee convergence to fairness
- Easy to show that "R_I , R_D = *constant*" does not converge
- Hence the "add/drop one layer" method does not converge to fairness

Overview (cont'd)

- Among linear functions R_I and R_D, only one method converges to fairness
- This method is called AIMD (Additive Increase, Multiplicative Decrease):

$$R_I(r) = \alpha,\ R_D(r) = \beta \cdot r$$

- AIMD is used in TCP (Transmission Control Protocol)

Overview (cont'd)

- *Binomial schemes* are a special case of l-D congestion control (proposed in 2001 by Bansal *et al.*):

$$\begin{cases} R_I(r) = \lambda r^{-k}, k \geq 0 \\ R_D(r) = \sigma r^{l}, l \leq 1 \end{cases}$$

- Powers (*k*, *l*) determine the shape of the increase and decrease curves
- Power *l* cannot be greater than 1 (otherwise, rates may become negative)
- Necessary condition for convergence is $k + l > 0$

Overview (cont'd)

- A scheme is *TCP-friendly*, if it achieves the same throughput as a TCP connection over a shared link
- AIMD is TCP-friendly
- Binomial schemes with $k+l = 1$ are TCP-friendly
- Two such TCP-friendly schemes were introduced in 2001:
  - IIAD (Inverse Increase, Additive Decrease) with $l = 0$, $k = 1$
  - SQRT (Square Root) with $l = k = 0.5$
- Even though two schemes may have the same throughput, they may differ in other characteristics
- One important characteristic is called *scalability*

Scalability of Congestion Control

- Scalability aspects of congestion control have not been extensively studied before or even quantified
- We created a new *measure of scalability* and use it to compare various congestion control schemes
- We define *scalability* of a scheme as "the increase in packet loss as a function of the number of flows *n* that use the scheme over a shared bottleneck"
- Suppose p_n is packet loss when *n* flows use a scheme
- Hence, scalability is described by the *packet loss increase factor* $s_n = p_n / p_1$

Scalability (cont'd)

- How does scalability matter?
- NACK-based protocols are very sensitive to the scalability aspects of a scheme, much more so than ACK-based protocols
- In ACK-based protocols, upon loss of communication, the sender simply stops, hence increasing stability of the network
- In the absence of feedback, congestion control in NACK-based protocols becomes "open-loop," or simply CBR:
  - If the communication from the receiver is delayed, the sender continues stressing the network at the same rate
  - Congestion control in NACK protocols is considered very difficult

Scalability (cont'd)

- To investigate the scalability of binomial algorithms, we use steady-state analysis and continuous fluid approximation
- We compute *n*-flow packet loss p_n:

$$p_n \approx \frac{\lambda^2 (k+2) n^{2k+2}}{C^{2k+2}\left(1-\left(1-\sigma(C/n)^{l-1}\right)^{k+2}\right)}$$

- *C* is the capacity of the bottleneck link
- Packet loss increase factor $s_n = p_n / p_1$ (describing the scalability of a particular scheme) is given by:

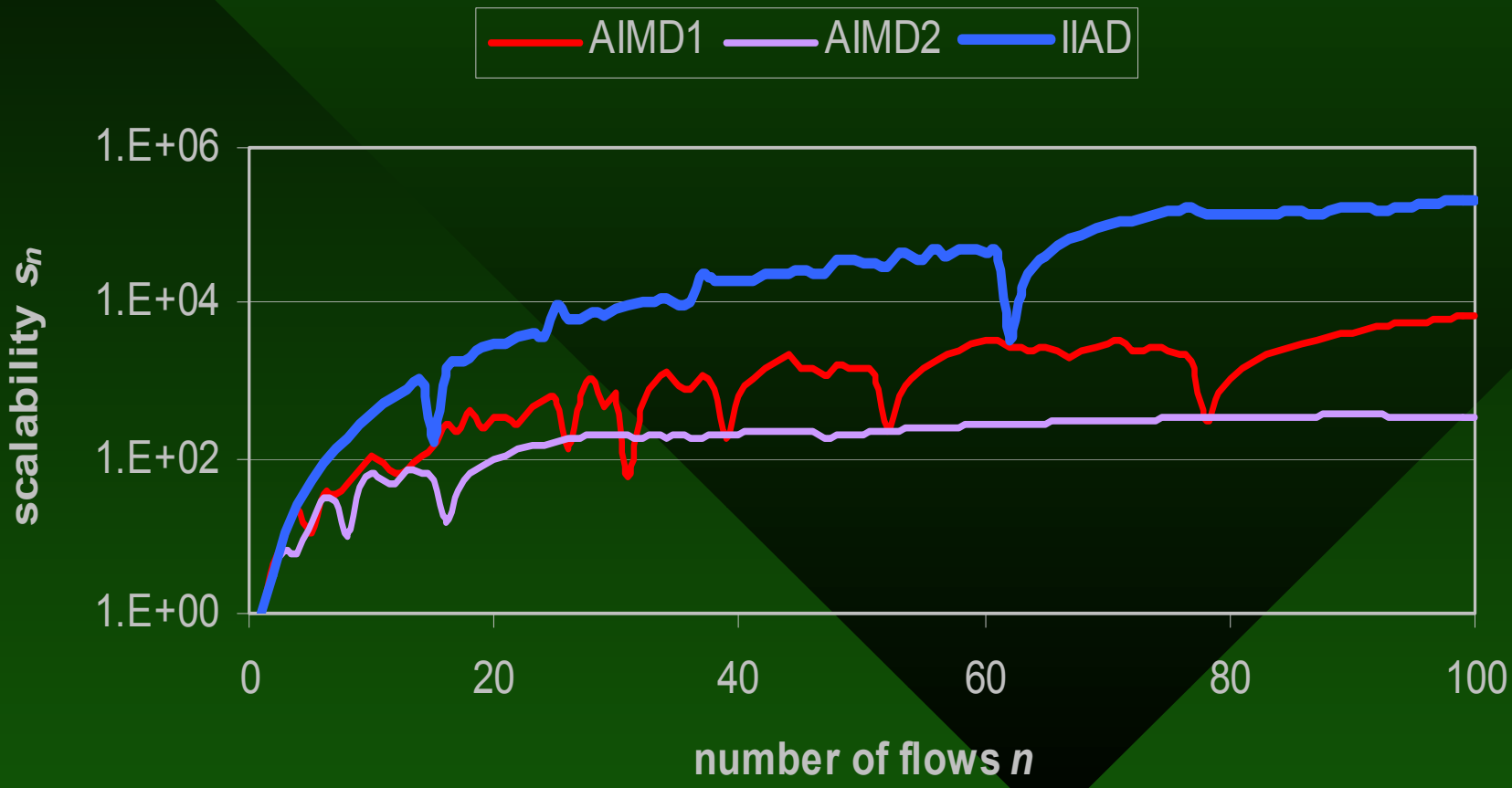
$$s_n \approx \frac{n^{l+2k+1}\left(2-(k+1)\sigma C^{l-1}\right)}{2-(k+1)\sigma(C/n)^{l-1}} = O(n^{l+2k+1}).$$

Scalability (cont'd)

- This formula can be used to compare the scalability of different schemes
- Schemes with the lowest scalability power $l+2k+1$ are best
- AIMD is $O(n^2)$
- IIAD is $O(n^3)$
- SQRT is $O(n^{2.5})$
- In practice, scalability is typically proportional to n^{l+2k}, but worse-case scalability n^{l+2k+1} does happen in certain scenarios

Scalability Examples

- Discrete event simulation below shows three flows under different conditions – AIMD1 $\propto n^{1.91}$(red), AIMD2 $\propto n^{1.27}$(purple), and IIAD $\propto n^{2.67}$(blue):



Scalability Examples (cont'd)

- Such high loss increases mean that these schemes will not be able to provide adequate performance once deployed globally

Scalability Limitations

- Among TCP-friendly schemes (i.e., $k+l = 1$), parameter s_n grows as $O(n^{3-l})$
- Hence, among all *TCP-friendly* schemes, the one with the largest l scales best
- Recall the restriction on binomial schemes that prevents the scheme from assuming negative rates is: $l \leq 1$
  - Example $l = 2$: new rate $r - \sigma \cdot r^2$ cannot be guaranteed to be positive for all $r > 0$, no matter how we select constant σ
  - This is due to the fact that r is considered "unlimited" (i.e., upper limit is not known a-priori)

Scalability Limitations (cont'd)

- Notice that among all TCP-friendly schemes, AIMD has the largest *I* equal to 1
- Hence, AIMD scales *best* among TCP-friendly schemes
- The only way to improve scalability beyond $O(n^2)$ is to somehow find a tight upper bound on sending rate $r(t)$
- One such upper bound is the *bottleneck capacity* of an end-to-end path
- The *bottleneck capacity* (or *bottleneck bandwidth*) is the speed of the slowest link of an end-to-end path

Scalability Limitations (cont'd)

- No prior work considered the bottleneck bandwidth in conjunction with congestion control
- Not only do we aim to improve the scalability of AIMD (i.e., $O(n^2)$), we also attempt to achieve "ideal-scalability" as explained below
- "Ideal scalability" is such choice of increase-decrease powers that ensures constant packet loss
- In other words, under ideal scalability, $p_n = p_1$ for all n

Ideal Scalability

- Recall that packet loss increases as $O(n^{l+2k+1})$
- For ideal scalability we want the power to be zero:

$$l+2k+1 = 0$$

- To maintain convergence to fairness, we need:

$$k+l > 0$$

- Combining both conditions above, we find that the necessary conditions of ideal stability are:

$$l > 1 \text{ and } k < -1$$

- Hence, we have the same restriction on l that can only be lifted with the knowledge of bottleneck capacity C

Ideal Scalability (cont'd)

- Assume that we know bottleneck capacity C
- Rate r is limited by a constant: $0 < r \leq C$
- We introduce a new class of *ideally-scalable congestion control* (ISCC) that maintains convergence while $I+2k+1$ equals zero

ISCC

- Knowing *C*, we can adjust constants (λ, σ) to each path so that rate $r(t)$ is never reduced below zero
- At the same time, we want to be able to "tune" the scheme to achieve different levels of packet loss and efficiency
- One such selection is given by the following:

$$\sigma = \frac{1}{m_D C^{l-1}}, m_D \geq l \qquad \lambda = \frac{C^{k+1}}{m_I}, m_I \geq 1$$

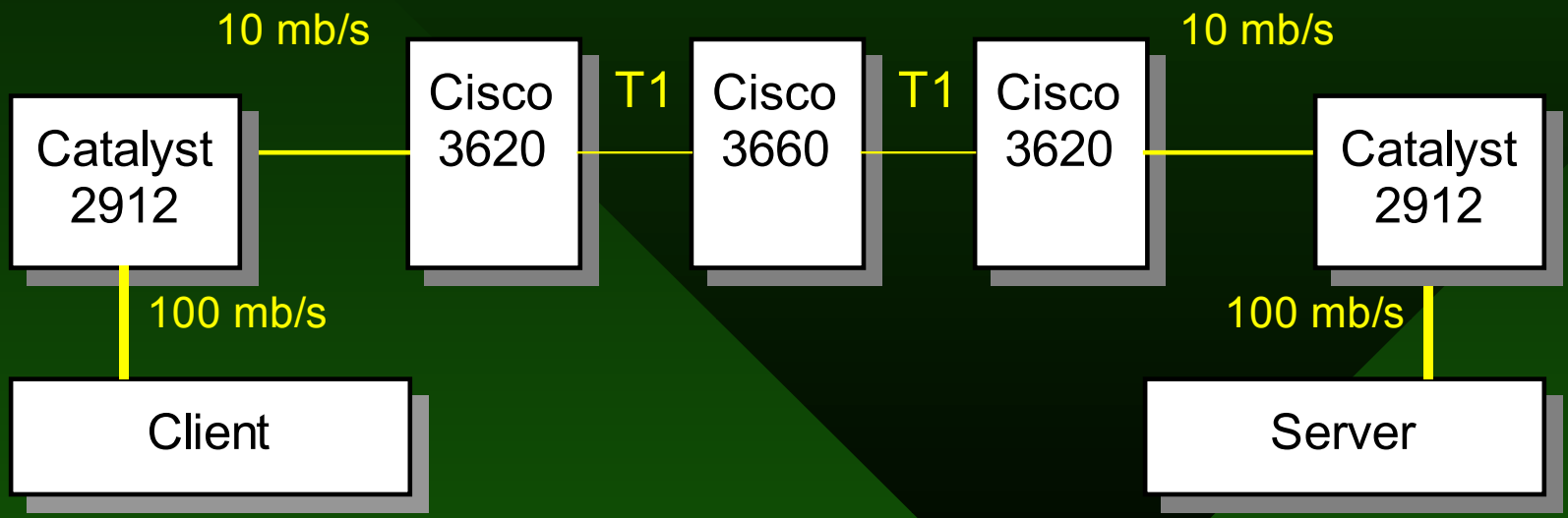
- Constant m_D is the *efficiency* factor, where higher values of m_D result in higher efficiency (i.e., higher link utilization)
- Constant m_I is the *aggressiveness* factor, where higher values of m_I result in less aggressive behavior (i.e., less packet loss)

Experiments

- Implemented NACK-based congestion control and binomial algorithms in our streaming software
- Special module controls the choice of congestion control (IIAD, SQRT, TFRC, ISCC, etc.)
- Tested between a Unix server and a Windows 2000 client, with the number of flows between 2 and 50
- 10 minute video sequence coded with MPEG-4 FGS:
  - 14 kb/s base layer
  - 1,190 kb/s FGS enhancement layer
- Each flow maintained a rate between 14 and 1,204 kb/s during the experiment

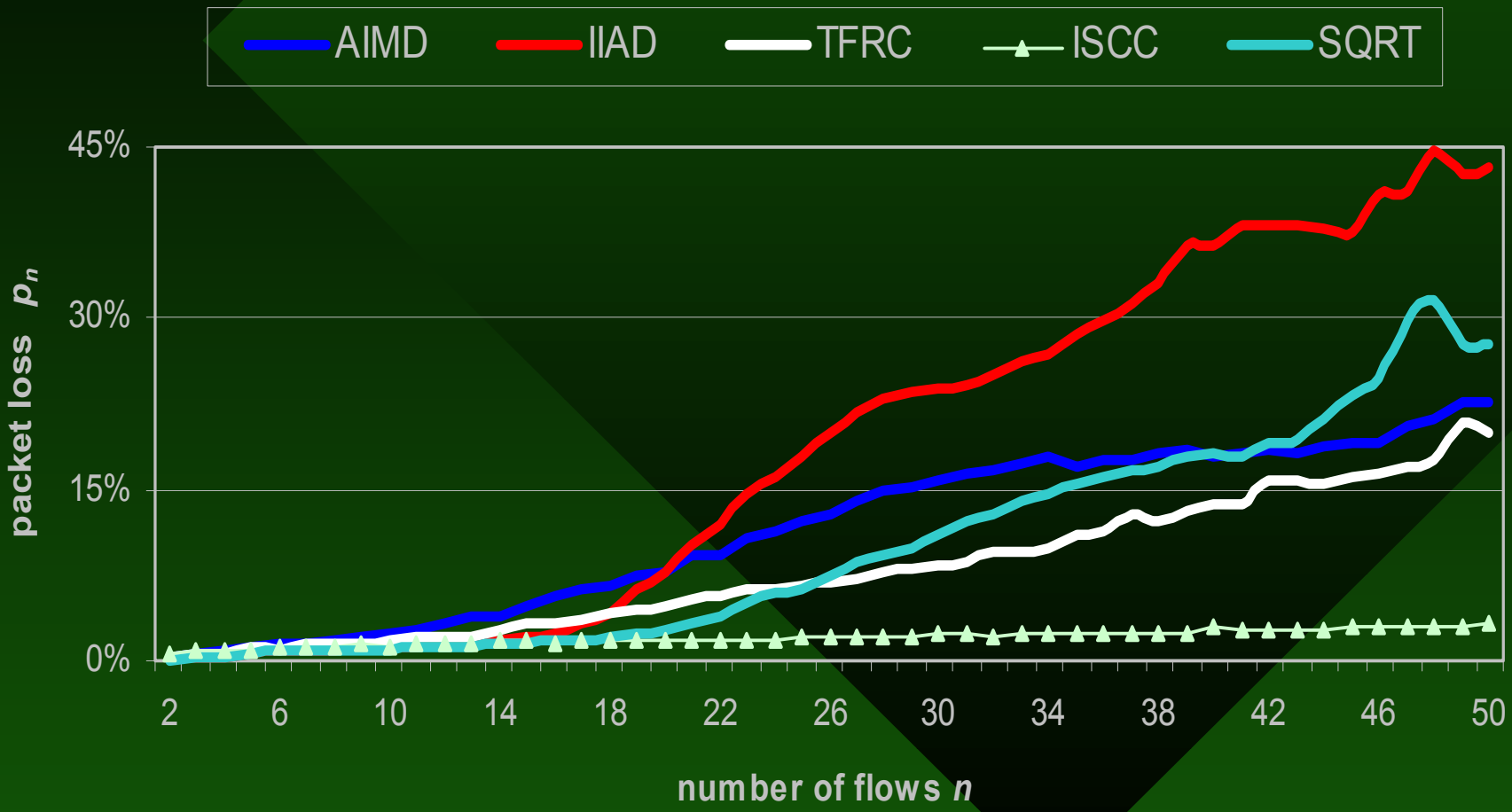
Experiments (cont'd)

- Setup including two high-speed ethernets on each side of the bottleneck T1 links (see below)
- WFQ and WRED (QoS features of routers) were disabled to reflect the current setup of Internet routers



Scalability Results

- The following figure shows packet loss p_n for all schemes as a function of n



Scalability Results (cont'd)

- All non-scalable schemes suffered from high packet loss rates:
  - as high as 45% in IIAD (p_2 = 0.29%, s_{50} = 151)
  - 32% in SQRT (p_2 = 0.10%, s_{50} = 268)
  - 27% in TFRC (p_2 = 0.27%, s_{50} = 74)
  - 22% in AIMD (p_2 = 0.38%, s_{50} = 59)
- ISCC maintained virtually constant loss at 3% (p_2 = 0.57%, s_{50} = 5.6)
- In non-scalable schemes, underflow events *in the base layer* were frequent, even given a large startup delay (3 seconds):
  - IIAD and SQRT maintained no picture for up to 66% of the time
  - AIMD and TFRC between 11 and 40% of the time

Scalability Results (cont'd)

- The ISCC scheme recovered all base-layer and FGS-layer frames before their deadlines
- This in fact represents an "ideal" performance for the end-user

Drawbacks

- However, ideal performance comes at a price:
  - All flows must acquire *consistent* estimates of the bandwidth
  - Ideally-scalable schemes are *too* TCP-friendly
  - In fact, ISCC will yield bandwidth to TCP due to its less-aggressive behavior
  - Differentiated Services (DiffServ) is required in the network
  - Efficiency is similar to that of AIMD, but convergence is slower
- The Internet is moving toward DiffServ
- Hence it is possible that UDP and TCP traffic will end up in different router queues
- ISCC is possible in DiffServ-enabled Internet

Congestion Control Conclusion

- Applications with *rate-based* flow control are very sensitive to what kind of congestion control is employed
- Hence, traditional schemes designed for ACK-based protocols are not well suited for NACK-based protocols
- At the same time, ACK-based flow control is poorly suited for real-time streaming
- Consequently, future real-time streaming protocols should only use congestion control that scales well in the presence of a large number of concurrent flows
- One such class of scalable schemes was developed in our work and is called ISCC