Inference of Flow Properties from Sampled Packet Streams

Nick Duffield, Carsten Lund, Mikkel Thorup

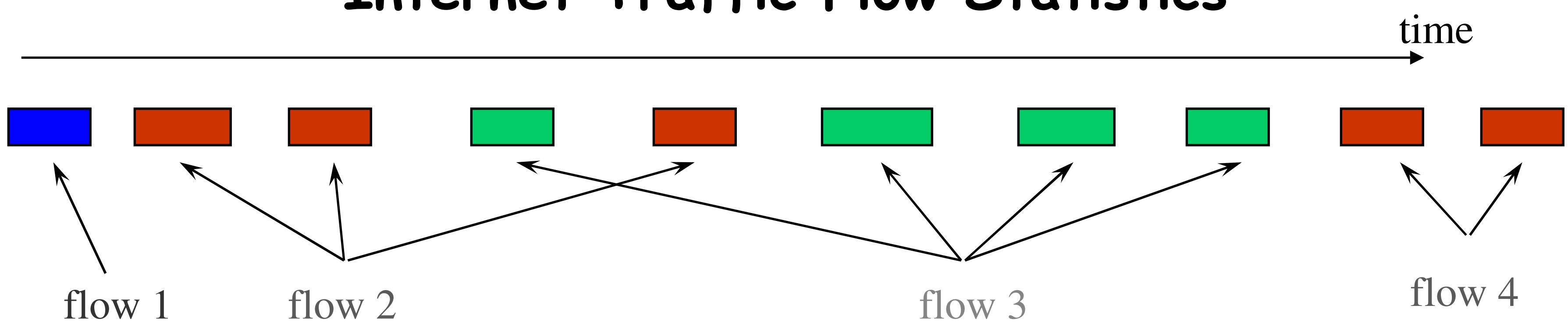
AT&T Labs Research, Florham Park

http://www.research.att.com/projects/flowsamp

General Setting

- Traffic measurements increasingly have to be sampled
  - Resource constraints under ever-increasing traffic speed, volumes
- Measurement-based measurement applications
  - Increasingly need detailed traffic properties:
    - resource usage differentiated by traffic class
    - traffic characteristics, composition
    - anomaly detection
    - spatial and temporal correlation of measurements
- Problem
  - How can these applications work effectively with sampled data?
- This talk:
  - inference of some properties of traffic flow statistics from samples

Internet Traffic Flow Statistics



- Measured Internet flows
  - set of packets with common property, observed in some time period
- Common property
  - "key": built from header fields (e.g. src/dst address, TCP/UDP ports)
- Flow termination criteria
  - interpacket timeout
  - protocol signals (e.g. TCP FIN)
  - ageing, flushing, …
- Flow summaries
  - reports of measured flows exported from routers
  - flow key, flow packets/bytes, first/last packet time, router state
- Measured flow semantics
  - artificial, may capture appl. transactions if good start/termination criteria

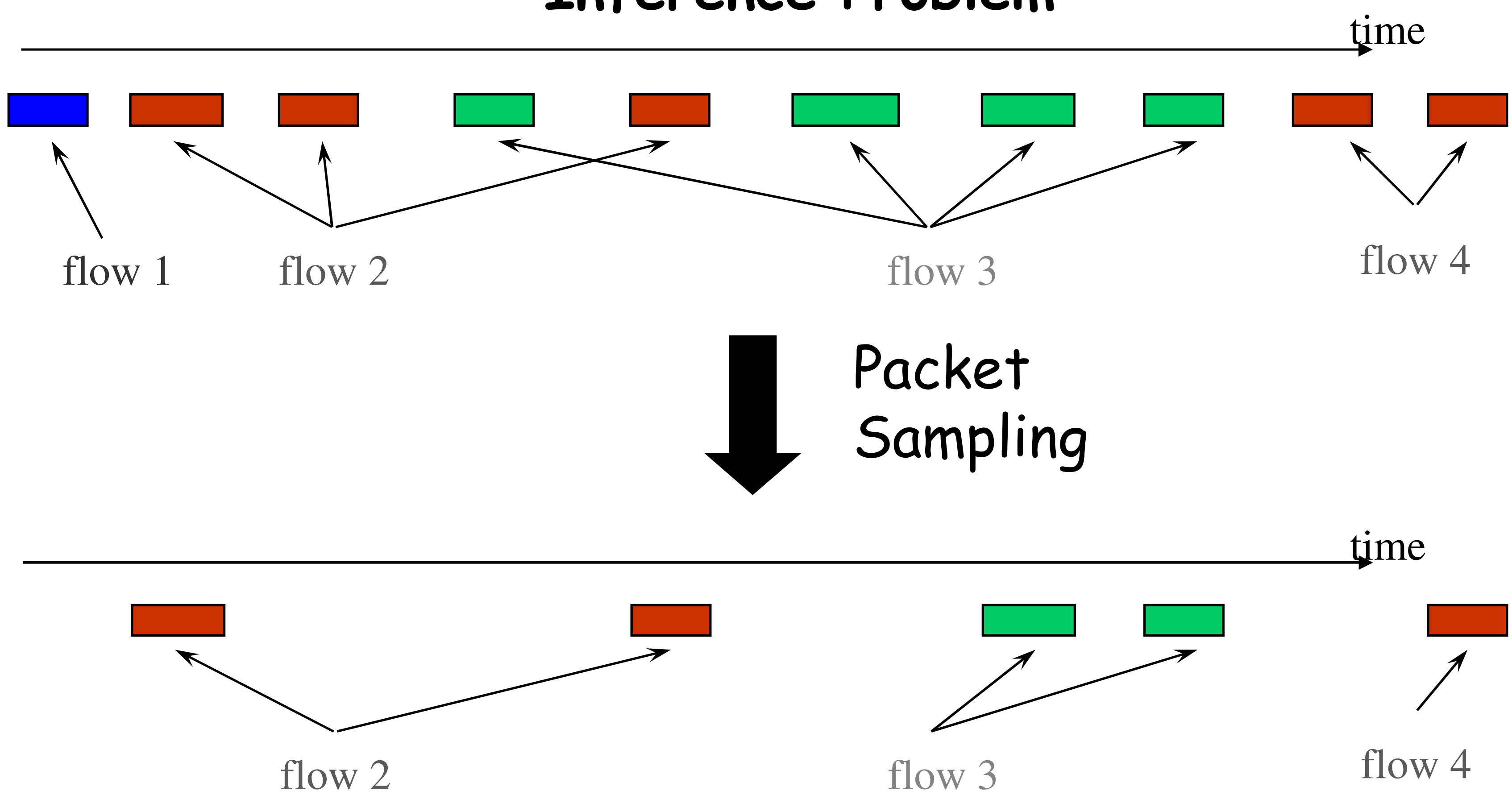
Need to Understand Detailed Flow Statistics

- Resource requirements in routers
  - number of concurrently active flows
- Resource requirements in measurement infrastructure
  - rate of production of flow statistics
- Traffic characterization
  - packet/byte rate of original traffic
  - rate of occurrence of original flows
  - average packet/bytes per original flow

The Need for Packet Sampling at Router

- Keep cache of active flows
  - for keys seen, but corresponding flow not yet terminated
- Packet classification
  - each arriving packet: cache lookup to match key
    - if match: modify cache entry, e.g., increment counters, adjust timers
    - else: instantiate new cache entry
- Cache resources for high end routers
  - memory: 1,000s of flows active simultaneously
  - speed: look up at line rate
  - hence cost: need lots of fast memory
- Packet sampling
  - form flows from sampled packet stream (e.g. 1 in N periodic)
    - will call these "packet sampled flows"
  - reduce effective packet rate
  - save cost: slower memory sufficient

Inference Problem



How to infer flow properties of **original** packet stream from flows of the **sampled** packet stream ?

Program

- Compare properties of packet sampled flows and original flows
  - rate of production of flow statistics
  - number of concurrently active flows
  - dependence on sampling rate, interpacket timeout
- Modeling, analysis, prediction
  - of packet sampled flow statistics, given original flows
- Inversion and Inference
  - recover properties of original flows
  - from packet sampled flow statistics

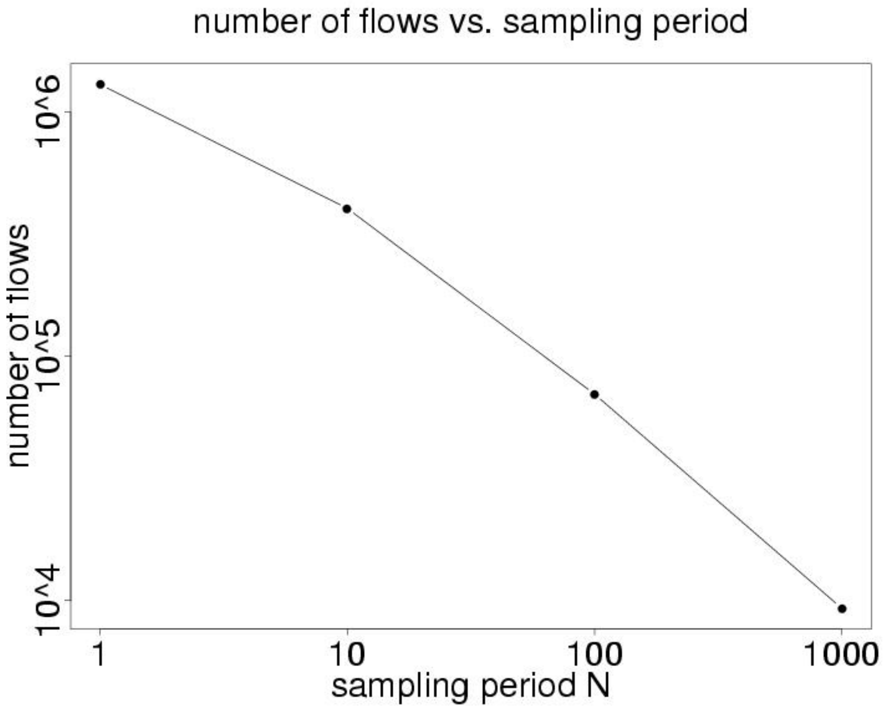
Resource Requirements:

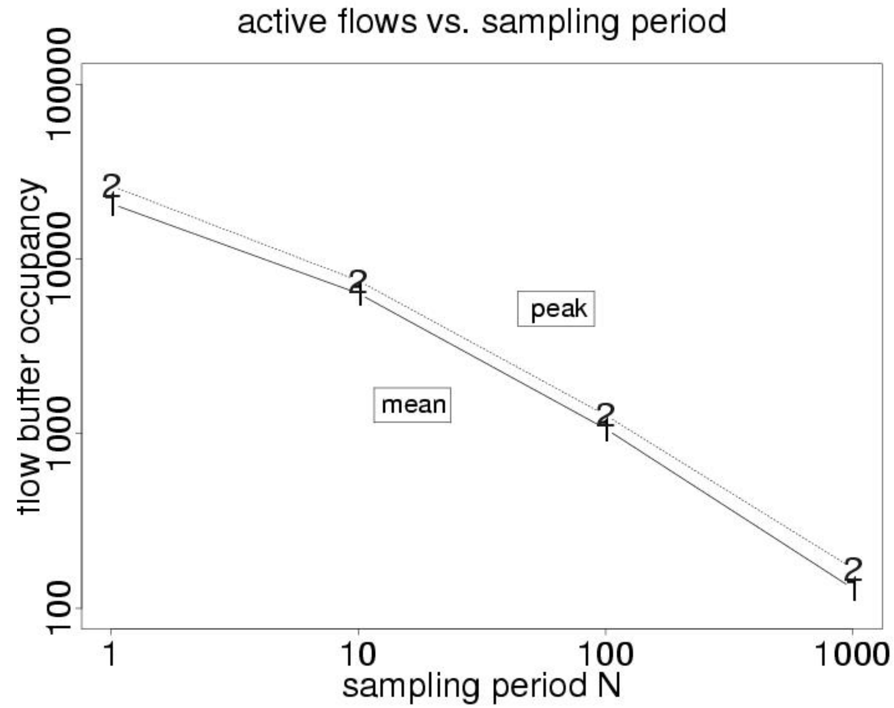
Experiments

Resource requirements: experiments

- Packet header trace
- Sample periodically 1 in N
  - call N the Sampling Period
- Form flow statistics
  - key = src/dst IP address + src/dst TCP/UDP port numbers
  - flow termination: interpacket timeout T
  - flow semantics
    - protocol based termination would be supressed under sampling
  - flow statistics
    - per flow: packets, bytes, duration
- Sensitivity
  - flow statistics almost insensitive to details of sampling process
    - compared 1 in N periodic, independent with probability 1/N
    - difference noticeable (barely) only if #active flows < N

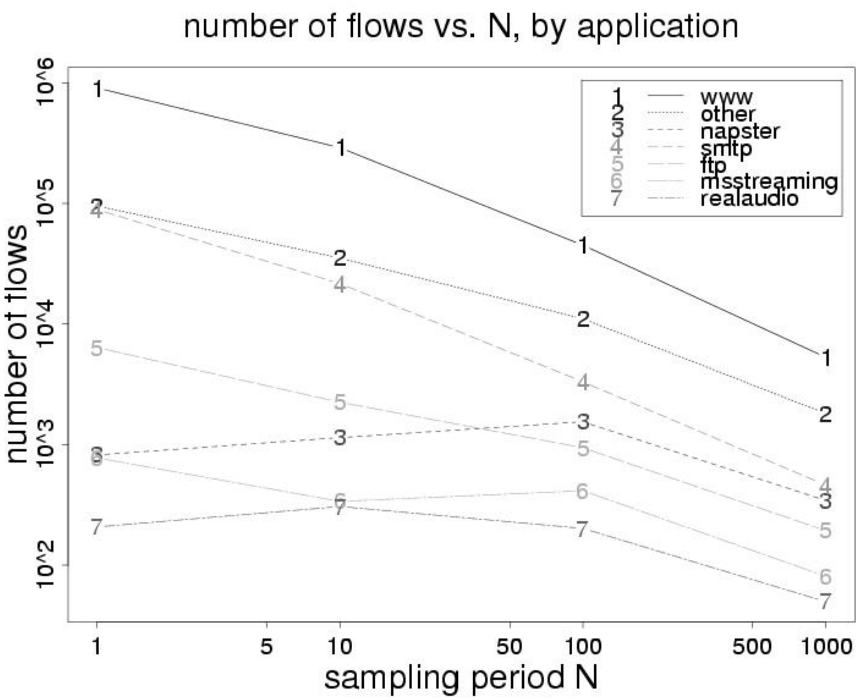
Rate and #active flows: aggregate traffic

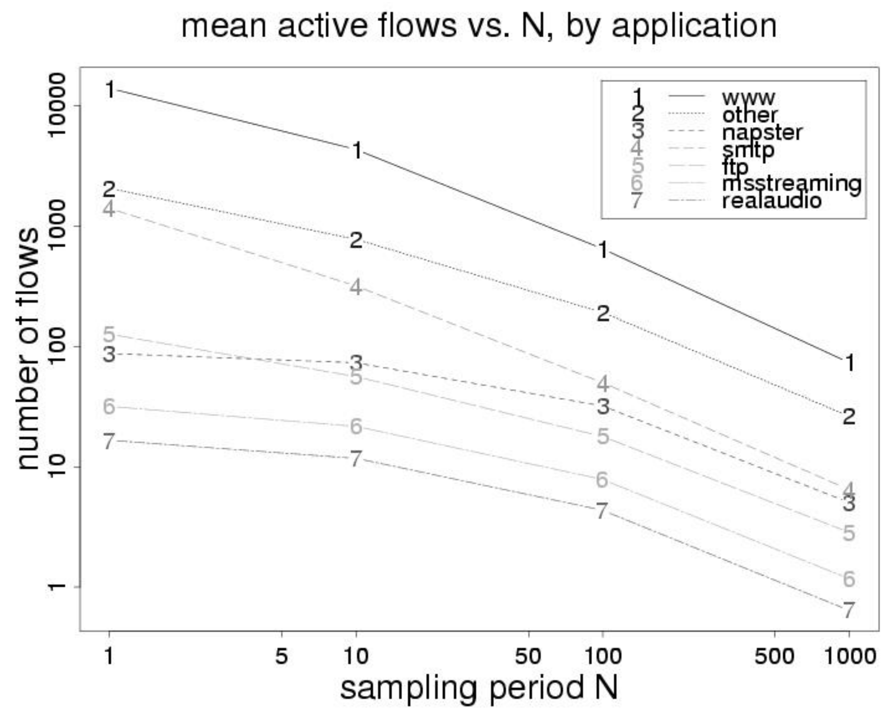




- Broad features
  - rate and #active flows decreasing,
  - expect eventually proportional to 1/N
    - probability to at least one of p packets ≈ p/N for large N
- Details vary from trace to trace
  - need to understand dependence on traffic constituents

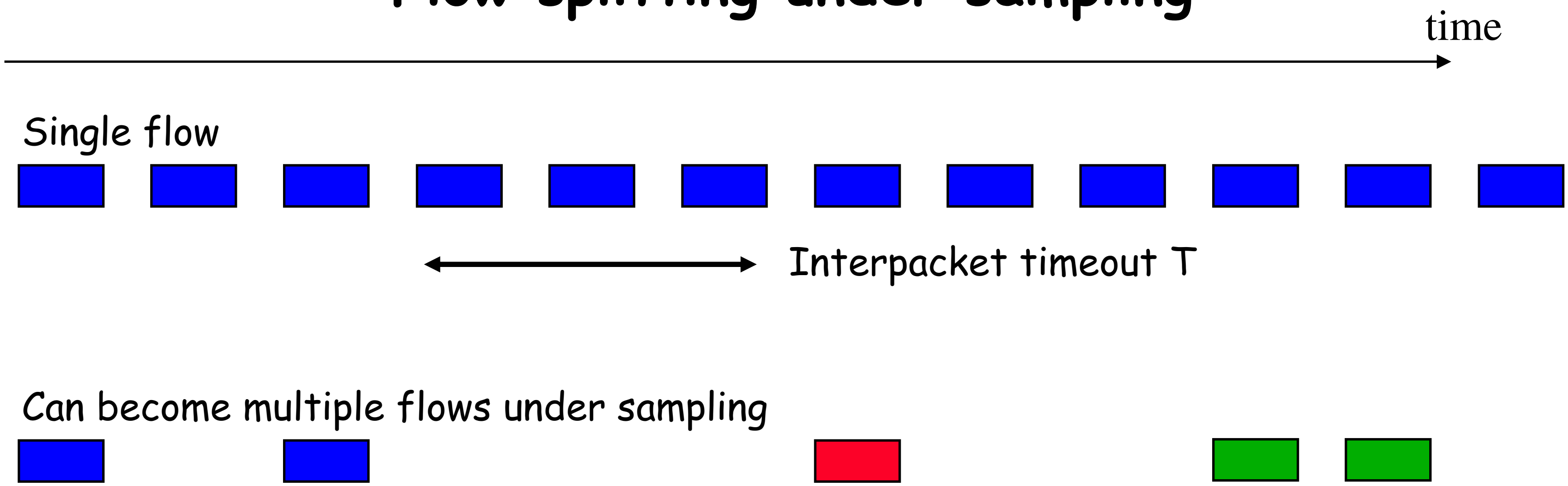
Rate and #active flows: by application





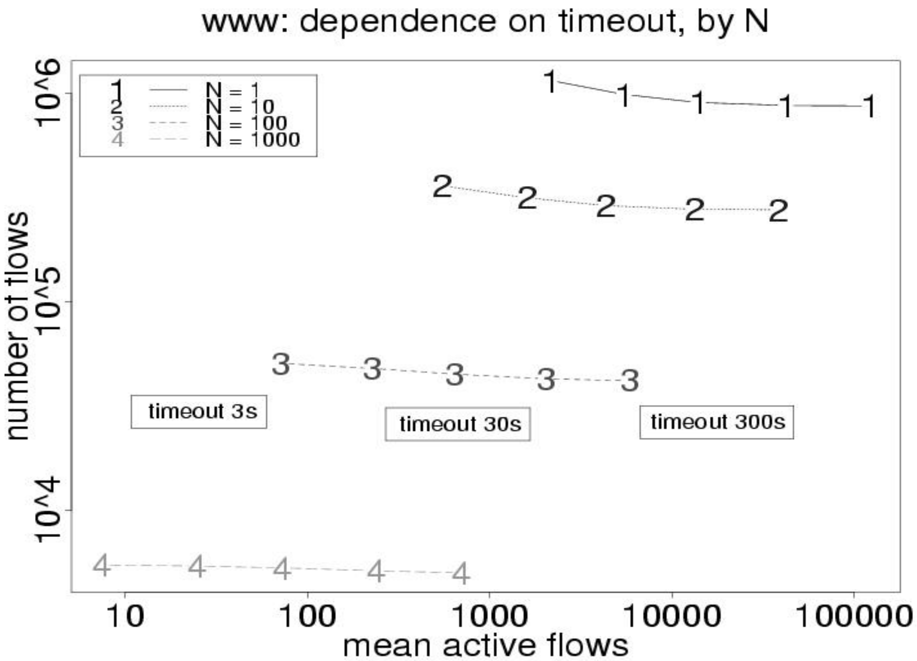
- Application identified by port number (well-known ports + custom)
- Rate of flow production
  - can **increase** with N for some applications, eventually decreasing
    - napster, ms-streaming, realaudio
- Mean active flows
  - decreasing with N, although slower for some applications: napster

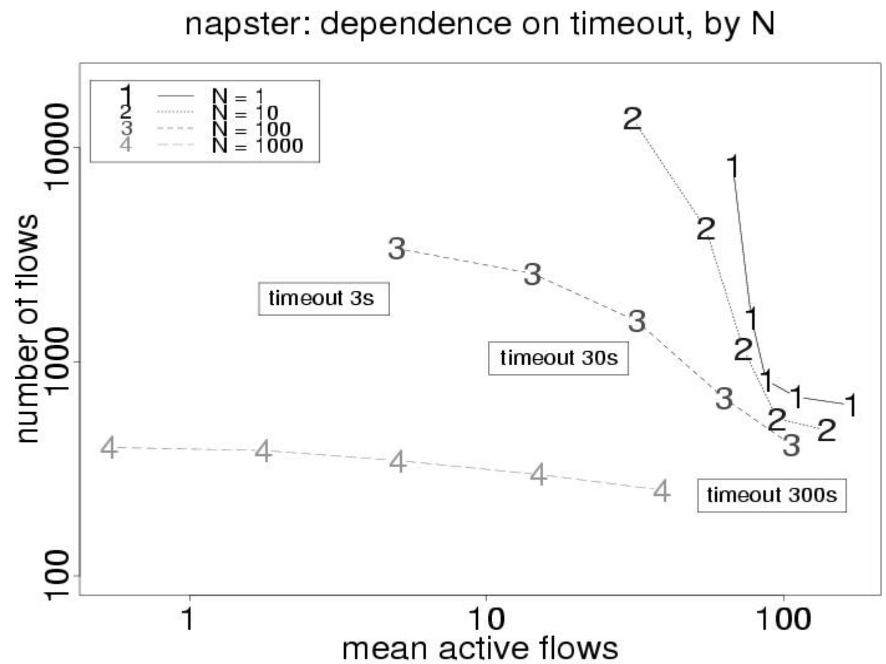
Flow splitting under sampling



- Sampling increases interpacket times
- Flow splitting when interpacket time exceed interpacket timeout
- Flows vulnerable to splitting: call these **sparse**
  - flows with many packets, not too fast packet rate
    - e.g. streaming, p2p applications
- Question
  - if increase T, as N increases: can we better maintain flow semantics?

Rates and #active flows: trade-offs





- Trade off: increase timeout T:
  - potentially less splitting: fewer measured flows, more active flows
- Left: non-sparse application (www: mean flow length 6 packets)
  - little flow splitting in any case
  - if larger T: roughly linear increase in active flows, flow rate roughly unchanged
- Right: sparse application (napster: mean flow length 455 packets)
  - smaller N: big trade off between rate and #active flows
  - larger N: trade-off washes out (typically only 1 packet sampled)

Sparseness and Flow Splitting:

Modeling

Make model of flow splitting

- Motivation
  - No simple black box model of rate and #active flows
    - based on just aggregate traffic rates, N, T
- Idea:
  - starting with set of original flow statistics
    - (n_i packets , b_i bytes , t_i duration) for flows i = 1,2,…, m, over duration D
    - from trace of collected flow statistics, or statistical model
  - use model to predict, given sampling period N, interpacket timeout T
    - mean rate of production of flow statistics
    - mean # concurrently active flows

duffield@research.att.com

Model of flow splitting: flow production rate

- Rough conditions for splitting of flow (n,b,t)
  - mean time between sampled packets exceeds timeout:
    - $N\,t/(n-1) > T$
  - more than one sampled packet on average
    - $n/N > 1$
  - say flow is *sparse* if both conditions hold
- Simple model: assume constant spacing of sampled packets
- Number of flows produced:
  - if sparse, expectation of n/N single packet flows
  - else get 1 flow
    - with probability 1 if $n > N$ (multi packet flow)
    - with probability n/N if $n < N$ (single packet flow)
- Wrap together: expect $f(n,t;N,T)$ flows on average,
  - $f(n,t;N,T) = 1$ if $(n-1)T \geq Nt$
    $= n/N$ otherwise
- Estimate flow production rate: $F = \Sigma_i f(n_i , t_i ; N, T) /D$

Model of flow splitting: #concurrently active flows

- Active duration:
  - if sparse, get n/N single packet flows,
    - each has cache open for duration T
    - total active time Tn/N
  - else get 1 flow
    - with probability 1 if n > N (multi packet flow)
      - expected active time T + time between first and last sampled packets
      - T + t(n-N)/(n-1)
    - with probability n/N if n < N (single packet flow)
      - expected active time Tn/N
- Wrap together: total average active time a(n,t;N,T)
  - $a(n,t;N,T) = T + t(n-N)/(n-1)$ if $(n-1)T \geq Nt$
$= Tn/N$ otherwise
- Estimate mean # concurrently active flows
  - total active time / duration D
  - $A = \Sigma_i \, a(n_i, t_i; N, T) / D$

Accuracy of prediction from model

- Compare
  - prediction: apply model to flow statistics of original traffic stream
  - experiment: form measured flows from sampled packet stream
- Ratios: predicted/experiment
  - flow production rate

N / T	10	100	1,000	10,000
1	**1.22**	**1.15**	**1.04**	**1.00**
10	**1.21**	**1.13**	**1.13**	**1.02**
100	**1.23**	**1.10**	**1.10**	**1.09**
1,000	**1.23**	**1.08**	**1.10**	**1.06**

mean #concurrently active flows

N / T	10	100	1,000	10,000
1	**1.18**	**1.08**	**1.01**	**1.00**
10	**1.21**	**1.13**	**1.08**	**1.01**
100	**1.23**	**1.11**	**1.10**	**1.05**
1,000	**1.23**	**1.09**	**1.10**	**1.05**

  - good, but better agreement for largest N (exceeding most flow lengths)
    - typically only one packet typically sampled, regardless of sampling details
- More complex model available:
  - flow packets independently distributed over flow duration
  - uniformly better agreement with experiment

Do we really need to model sparseness?

- Yes! Compare with model with no splitting
- Ratios: predicted/experiment
  - flow prediction rate

N / T	10	100	1,000	10,000
1	**0.89**	**0.66**	**0.64**	**0.73**
10	**1.08**	**0.79**	**0.65**	**0.69**
100	**1.17**	**0.96**	**0.86**	**0.77**
1,000	**1.23**	**1.04**	**1.01**	**1.00**

mean #concurrently active flows

N / T	10	100	1,000	10,000
1	**1.40**	**4.24**	**13.3**	**57.4**
10	**1.31**	**2.23**	**7.10**	**14.5**
100	**1.43**	**1.99**	**2.53**	**2.82**
1,000	**1.25**	**1.19**	**1.24**	**1.18**

  - generally worse agreement that with sparseness modeled
  - particularly bad for #active flows, large N
- If splitting ignored:
  - underestimate rate of flow production (fewer measured flows)
  - overestimate # concurrently active flows (ignore inactive time between split flows)

Inferring original flow statistics from packet sampled flow statistics

Characteristics of Interest

- Motivation
  - assume only packet sampled flow statistics are available
  - want to determine characteristics of original flows
- Which characteristics?
  - original packet/byte rates
  - arrival rate of original flows
  - average packets and bytes per original flow
- Why might this be difficult?
  - some original flows are missed altogether: no packets sampled
- Trick:
  - supplement with protocol level information, when available

Some easy estimates: usage

- Network usage: packet an/or bytes in original packet stream
  - or differentiated per traffic class
- Model: packets independently sampled with probability 1/N
- Estimates:
  - # original packets P by P_{est} = N * # sampled packets
  - # original bytes B by B_{est} = N * # sampled bytes
- Properties (Bernoulli sampling):
  - unbiased estimators: $E[P_{est}] = P$; $E[B_{est}] = B$
  - standard error bounds
    - packets: $std\_dev(P_{est})/P \leqq sqrt(N/P)$
    - bytes: $std\_dev(B_{est})/B \leqq b_{max} / b_{av} \, . \, sqrt(N/B)$
      - b_{max} = maximum packet size
      - b_{av} = average packet size

duffield@research.att.com

Estimating number of original TCP flows

- How to estimate number M of original TCP flows?
- Use trick for TCP flows reported by Cisco NetFlow
  - packets of TCP connection carry flags (bits) for control
  - first packet of a TCP connection carries the SYN flag
  - flow statistics include cumulative OR of its packets' code bits
  - hence can tell whether TCP flags were set in at least one flow packet
- Model (SYN flags in TCP flows are well-behaved)
  - each original TCP flow contains exactly one SYN packet
    - expect close adherence to model, modulo retransmits, packet drops
    - experiments
      - long flow traces: very rare for flow to have no SYN packet
- Estimation
  - each SYN packet sampled with probability 1/N
  - estimate: M_1 = N * #{sampled flows with SYN flag set}
  - properties: unbiased estimator of M = #{original TCP flows}
    - under the model assumptions

Estimating number of original TCP flows (2)

- Estimator M_1 : uses only sampled SYN flows
- Decrease estimator variance by using all flow statistics?
- Basis: estimate number of flows M_0 that were not sampled at all!
  - Let N_0 = (N - 1) * #{flow has only SYN sampled}
  - Theorem: under model assumption
    - $E[N_0] = E[M_0]$
  - Proof: consider event S that flow has no non-SYN packet sampled
    - {flow not sampled} = {SYN not sampled} $\cap$ S
    - {only SYN sampled} = {SYN sampled} $\cap$ S
  - Hence, since {flow not sampled}, {only SYN sampled} $\subseteq$ S,
    - Prob[flow not sampled] = Prob[S] (1-1/N)
    - Prob[only SYN sampled] = Prob[S] / N
  - Prob[flow not sampled] = (N - 1) * P[only SYN sampled]

Estimating number of original TCP flows, byte/packets per flow

- Consequences
  - if there were no flow splitting:
    - #{measured flows} = #{original flows with ≥1 packetsampled}
  - $M_2 = N_0$ + #{sampled flows} is unbiased estimator if no flow splitting:
    - $E[M_2]$ = E[#{unsampled flows}] + E[#{sampled flows}] =#original flows
- Comparison
  - M_1: higher variance (less data), unbiased by flow splitting
  - M_2: lower variance (more data), biased by flow splitting
- Corresponding estimates of mean packets per flow, bytes per flow
  - packets: $p_{est,\,i} = P_{est} / M_i$; bytes: $b_{est,\,i} = B_{est} / M_i$; $i = 1,2$

Estimation Accuracy

- Restricted packet trace:
  - select only packets in original TCP flows starting a SYN packet

Mean length of original flows

N	$p_{est,1}$	$p_{est,2}$	StdErr	M_1
1	23.0	23.0	n/a	299875
10	22.4	22.1	0.12	307670
100	22.5	21.9	0.44	306400
1,000	22.0	21.7	1.23	313000

- Error comparable with standard deviation, but some bias
  - 7 times std_dev for N =10, < 1 std_dev for N=1,000
  - M_1 increases: small number of flows with more than 1 SYN packet
- Can improve accuracy of $p_{est,2}$ by scaling $T \propto N$
  - suppress splitting of sparse flows
- $p_{est,1}$ gives best accuracy

Summary

- Resource usage to form packet sample flows
  - sensitive to detailed traffic characteristics
  - developed simple model to predict from traces
- Inference of original traffic characteristics
  - from packet sampled flow statistics
  - bytes and packets: simple estimators, error bounds
  - number of original flows:
    - TCP flow only: use reported statistics of sampled SYN packets
    - estimator: good experimental accuracy
- One instance of a general problem:
  - to understand impact of sampling on measurement based applications