End-To-End Performance Guarantees in Computer Networks Using NetLets

Nageswara S. V. Rao
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831

April 14-15, 2000
Computer Science Symposium
Louisiana State University
Baton Rouge, LA

Research Sponsored by
Information Technology Office
Defense Advanced Research Projects Agency
Outline of Presentation

- Present-day Networks - Need for QoS
- NetLets: Basic Concepts
- Distributed Computing Environments
- Analytical Model and Performance Guarantees
Present-Day Networks: Internet

Very little Quality of Service (QoS) Guarantees:

- Internet packets are routed according to best-effort mechanism
- All packets are the same - control packet from PC to a waiting teraflop machine
- Once packet is sent, very little can be done about how it is sent

QoS is Needed Now:

- Large distributed simulations
- Data transfers to and from high performance machines
- Control of sensors and robots over networks

State and configuration of network must be exploited to achieve best performance

What type of QoS is Desired:

1. End-to-end guarantees on delay, jitter, etc., for various types of messages

2. Must be provided in a transparent manner to the application programmer
NetLets: Basic Idea

Currently, no control once data reaches the network

Netlets use both configuration and state of network.
Distributed Computing Over IP Networks

At Present:
- process-to-process communication is achieved as peer-to-peer mechanism
- parallelism in network is not taken advantage of
  e.g. congestion on a link is not bypassed

NetLets Offer Natural Solution:
Increase flow in less congested routes
- Use processes to assist in networking
NetLets for Distributed Processes

ORNL laid initial foundations for NetLets

Process-process communication is handled through Netlets that:

- estimate link statistics (non-linear estimators)
- compute “best” paths

Provide probabilistic end-to-end guarantees:

- distribution-free under stationarity conditions
- detailed probabilistic models are not needed: measurements are often sufficient
Distributed Computing over IP

We showed

that measurements are sufficient to provide end-to-end delay guarantees

- No need for extensive modeling

We derived probabilistic end-to-end delay guarantees for very general classes of networks

NetLets became possible as a result of unique combination of statistical estimation, graph and flow algorithms, and network engineering
Network Measurements

- Distributed environment consisting of four sites
- TCP/IP end-to-end delivery times vs message size
Routing Problem

**Given:** computer network $G = (V, E)$
available bandwidths $b(e)$, for link $e \in E$
link-delays $d(e)$, for link $e \in E$
queuing delay $q_v(r)$, for node $v \in V$, for message size $r$

**Message Transmission Problem:**
Compute a path to send message of $r$ units
from $s$ to $d$ with minimum end-to-end delay

**Simple Path:** $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$:
End-to-End delay for message size $r$:

$$t(r, P) = g \left( r, \min_{j=0}^{k-1} b(e_j) \right) + \sum_{j=0}^{k-1} d(e_j) + \sum_{j=0}^{k-1} q_{v_j}(r)$$

where $e_j = (v_j, v_{j+1})$;
$g \left( r, \min_{j=0}^{k-1} b(e_j) \right)$ is delay due to bandwidth;
$\sum_{j=0}^{k-1} d(e_j)$ is delay due to link-delays; and
$\sum_{j=0}^{k-1} q_{v_j}(r)$ is the queuing delay.
In practice, 
$g(.,.)$ and $d(.)$ can be accurately estimated. 
— they depend on links 
$q(.,.)$ — queuing delays are hard to estimate 
— they depend on other messages

— Message of size $R$ arrives at the source 
  according to an unknown distribution $P_R$
- At any node $v$:
  $Q_v$: queuing delay distributed according to unknown $P_{Q_v}$
  $R_v$: message size distributed according to unknown and $P_{R_v}$

Measurements:
$(Q_{v1}, R_{v1}), (Q_{v2}, R_{v2}), \ldots, (Q_{vn}, R_{vn})$
independently and identically distributed (iid) according to unknown $P_{Q_v, R_v}$

Fundamental Question:
When only measurements are available, 
can any guarantees be given on end-to-end delay?
**Optimal Paths**

**End-To-End Delay of** \( P \) in transmitting a message of size \( R \):

\[
T(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} Q_{v_j|R}
\]

**Expected Delay of** \( P \)

\[
\bar{T}(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} \int Q_{v_j} dP_{Q_{v_j}|R}
\]

**Best Expected Path:**

\( P^*_R \): path with minimum expected end-to-end delay,

\[
\bar{T}(P^*_R, R) = \min_{P \in \mathcal{P}} \bar{T}(P, R)
\]

where \( \mathcal{P} \) is set of all paths between \( s \) and \( d \).
Regression-Based Paths

**Empirical End-To-End delay**: Based on the estimator $\hat{q}_v(.)$

$$\hat{T}(P, R) = g(R, b(P)) + d(P) + \sum_{j=0}^{k-1} \hat{q}_{v_j}(R)$$

---

- $\hat{T}(.)$ can be computed since it involves only the measurements

**Best empirical end-to-end delay path**:

$$\hat{P} = \arg \min_{P \in \mathcal{P}} \hat{T}(P)$$

---

- Computed using our algorithm in $O(m^2 + mn \log n + nf(l))$ time
  - $f(l)$: cost of computing regression at $r$
Performance Guarantees

Compute $\hat{P}$ based on vector space method based entirely on measurements, such that for sample size

$$l = \frac{8192n^4\tau^2}{\epsilon^4} \left[ d\ln \left( \frac{512\epsilon r n^2}{\epsilon^2} \ln \frac{512\epsilon r n^2}{\epsilon^2} \right) + (n + 3) \ln 2 + \ln(n/\delta) \right],$$

we have

$$P \left\{ E_R | T(\hat{P}_R, R) - T(\hat{P}^*_R, R) | \geq \epsilon \right\} \leq \delta,$$

where $\sup \limits_\nu Q_\nu \leq \tau$

Informally, with high probability $1 - \delta$
expected delay of $\hat{P}$ is within $\epsilon$ of optimal expected delay, irrespective of underlying distributions.
Three Classes of NetLets

• **Distributed Computing in IP Networks**
  - integrate NetLets into native environments
  - thorough analysis of regression methods and sample sizes
  - path-tables can be used to quickly retrieve paths for given message size

• **ATM Networks**
  - bandwidths and delays are deterministic
  - routes must be reserved which adds random components

• **Active Networks**
  - routing code can be sent with messages
  - data-collection programs can be more flexible
Conclusions

- NetLets have potential for networking and network applications
  - offer a capability no one else is able to yet

- **Unique Combination** - statistics, algorithms, network engineering
  - NetLets are made possible by a unique synergy

- **Need to advance theory and implementation to the next level**
  - testing on benchmark problems