Software Performance Modeling of a Frame Relay Access Device

ADRIAN CONWAY

GTE Internetworking
(work carried out at Racal-Datacom)

First International Workshop on Software and Performance
Santa Fe, New Mexico
Oct. 1998
Frame Relay Network Access Device

• **Frame Relay Network**
  – virtual-circuit service
  – connects remote sites
  – economical compared to a private leased-line network

• **FRAD**
  – interconnects LANs and DTEs to the frame relay network
  – a multi-protocol multi-function device
Racal’s *FastFrame 600*
FRAD Protocol Architecture
FastFrame 600 FRAD Software

- Protocol Modules
  - standardized
    - acquired from various vendors
  - proprietary
    - different software development groups

- Protocol Integration with UNIX STREAMS Facilities
  - kernel routines for layered protocol software
  - modular architecture
    - drivers, modules, multiplexors, queues
  - simplifies development
  - reduces development time
Need for Software Performance Modeling and Analysis

- protocol modules developed by different groups
- performance of integrated system is unknown \textit{a priori} 
  - throughput, delay, burst handling
- shortened time to market 
  - less time for performance measurement, re-design, tuning
- performance ‘disasters’ at end of development cycle are costly 
- real-time performance is of increasing importance (e.g. voice/packet) 
- product requirements include performance 
- need UP-FRONT analysis at product specification phase 
  - architectural choices, verify design for performance requirements, expose potential flaws
- analysis at testing stage 
  - tool to help in parameter optimization
  - evaluate ‘last-minute’ changes
FRAD Performance Modeling and Analysis

- We propose a software performance model for data-networking products based on STREAMS

- UNIX STREAMS maps naturally into a queueing model
  - model focuses on data-transfer phase
  - exploit structure imposed by STREAMS

- service times (path-lengths) obtained from code measurements

- analysis using simulation or analytical techniques
STREAMS Modeling

user process

stream head

Multiplexor

module

driver

user space

stream head

kernel space

module

driver

hardware
messages (packets)

queues in a module

queues in a multiplexor
• **Message priorities in a STREAMS queue**
  – normal messages
  – expedited messages (levels 1 to 255)
  – high-priority messages
  – FIFO scheduling within each priority band

• **Message passing from one queue to another in STREAMS**
  – involves kernel routines
    • `putnext()`
    • `put()`
    • `putq()`
    • `service()`
(1) queue A service calls putnext

(2) putnext passes message to queue B put

(3) queue B put processes message

(4) put passes message to putq

(5) putq places message on queue B

(6) putq schedules queue B service
• **Scheduling of Service Routines by STREAMS**
  – service routines called by STREAMS scheduler
  – STREAMS scheduler is FIFO
  – STREAMS scheduler processes all messages on a queue when service routine is called

• **Inter-Queue Flow Control in STREAMS**
  – counter `q_count`
  – high and low water marks
  – service routines “put to sleep” if flow control in force
  – service routines “woken up” when flow control removed
State-Dependent FIFO Queueing Model

- Data traffic
- Creation of service routines
- State of STREAMS queues
- Service routines
- STREAMS scheduler
Model Analysis

• Analytical
  – difficult due to complex state-dependencies
  – possible to develop a simplified Markov chain model
  – a challenging performance analysis problem

• Simulation
  – simulation model implemented using OPNET Modeler
  – could automate building of OPNET model using OPNET EMA interface
Concluding Remarks

• Software performance modeling and analysis is an essential for data-networking product development

• research needed for SPE of data-networking products

• automated tools are needed for SPE

• we proposed a queueing model for SPE of STREAMS-based data-networking products