PROVISIONING AND ANALYSIS OF APPLICATION-SPECIFIC INTERNET MEASUREMENTS

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the
Graduate School of The Ohio State University

By

Prasad Calyam, B.E., M.S.

* * * * *

The Ohio State University

2007

Dissertation Committee:

Prof. Eylem Ekici, Adviser
Prof. David Lee
Prof. Steven Biblyk

Approved by

Adviser
Graduate Program in
Electrical and Computer Engineering
ABSTRACT

Real-time applications such as videoconferencing and Grid computing are being widely deployed on the Internet. To effectively meet the *Quality of Experience* (QoE) expectations of end-users, Internet Service Providers (ISPs) are increasingly relying on automated frameworks. These frameworks use Internet measurements pertaining to *Quality of Service* (QoS) metrics (e.g., delay and loss) and estimation techniques such as forecasting for initial selection and dynamic adaptation of resources. The degree of success achievable by these frameworks is being limited by the inability of ISPs to suitably provision and analyze Internet measurements. In this dissertation, measurement scheduling techniques are presented that provision QoS metrics on the Internet with the accuracy, regulation, frequency and flexibility desired by the frameworks. Also, a tool is presented that analyzes provisioned QoS metrics to estimate end-user QoE in real-time without involving actual end-users, a key requirement for framework automation.

Provisioning QoS metrics using measurement scheduling involves careful orchestration of active measurement tools. Since these tools consume non-negligible amount of network resources, it is essential to schedule them in a regulated and non-conflicting manner. In addition, QoS metrics need to be sampled with strict *periodicity* for accurate network weather forecasting. To address these measurement requirements, offline/online measurement scheduling algorithms are described. Simulations show that our algorithms can improve the schedulable utilization of offline measurements by 300% and the response time...
of online measurements by 50%. To evaluate their utility for performance forecasting, the algorithms are implemented in the popular Network Weather Service (NWS) framework. From the implementation experience, it is shown that no measurement scheduling algorithm that avoids measurement conflicts can cater strict periodicity. However, it is demonstrated how a piece-wise linear interpolation technique used with the algorithms can improve the forecast accuracy by reducing the mean square error by half. Lastly, an active measurement tool called Vperf is developed that uses provisioned network QoS measurements to estimate real-time QoE of applications, specifically involving Voice and Video over IP (VVoIP). Vperf uses a novel psycho-acoustic/visual cognitive model called GAP-Model. Validation experiments show that online Vperf measurements have matching performance when compared with time-consuming offline Peak Signal-to-Noise-Ratio (PSNR) measurements.
Dedicated to my Parents...
ACKNOWLEDGMENTS

As a child, I remember looking at the Ph.D. dissertations of both my parents in our house library and wanting to write one of my own. Now that I have written one myself, I would like to thank several people who have made it possible for me to realize my childhood ambition.

First, I would like to thank my current adviser Prof. Eylem Ekici and former adviser Prof. Chang-Gun Lee for all their guidance and inspiration. Both of them have led me in taking a systematic and critical approach to identify and solve research problems that matter. They have constantly encouraged me to set higher expectations for myself than I would normally, and thus have brought out the best of me in my work. I feel very fortunate for having the privilege to work closely with them and gaining numerous skills that I will surely benefit from for the rest of my career.

I am also grateful to Prof. David Lee and Prof. Dong Xuan. They have taken time out of their busy schedules on many occasions to discuss my research progress and have provided valuable feedback. In addition, attending their classes allowed me to learn several advanced networking concepts that were useful in my work. I am also thankful to Prof. Steven Bibyk for his constant encouragement and the concrete feedback he provided after patiently reading this dissertation.

Given that all my Ph.D. work was done while I was a full-time staff member at OAR-net/Ohio Supercomputer Center, I cannot thank enough all my colleagues. In particular,
I wish to thank my managers over the years: Paul Schopis, Pankaj Shah, Dr. Steve Gordon, Dr. Ashok Krishnamurthy, Dr. Stan Ahalt and Dr. Dave Hudak for the freedom they gave me to manage my work schedules. Their trust and confidence gave me the strength to meet my job expectations, while still making progress on my dissertation. In fact, they provided resources and staff time that enabled me to setup up the network testbeds used in my research. I also wish to acknowledge the support of my other colleagues: Arif Khan, Weiping Mandrawa, Mark Fullmer, Terry Lewis, James Rader, David Kehrle, Jason Poteat, Kevin Gecsi, Ananth Devulapalli, Siddharth Samsi, Christine Dorsey, Denis Walsh, Nancy Drugan Koehler, Julie Miller, Kelly Sitz, Connie Martin and Dana Rogers. I wish to thank my student colleagues who helped me with some of the testbed setup and software development chores: Mukundan Sridharan, Dima Krymskiy, Phani Kumar Arava, Mark Haffner, and Nathan Howes. I would like to make a special mention and express my deep gratitude to Dr. Bob Dixon. He introduced me to the fascinating world of Internet videoconferencing and also referred me to OARnet/OHio Supercomputer Center.

There are several others outside OARnet/OHio Supercomputer Center who have helped me in sowing the seeds of ideas that have contributed to my research growth. I would like to thank these extraordinary people for their time, interest and patience during their collaborations with me: Dr. Loki Jorgenson, Dr. Khaled Salah, Dr. Andrei Sukhov, Dr. Eric Boyd and Jeff Boote.

My job and research work at OARnet/OHio Supercomputer Center has been supported by several funding agencies. I express my sincere thanks to David Barber (The Ohio Board of Regents), Janet Poley (American Distance Education Consortium) and Cheryl Munn-Fremon (Internet2) for believing in my abilities and funding my research projects.
On the personal side, I would first like to thank my parents for all their love and support. Their words of wisdom have guided me in every step of the way. I also thank my sisters Lalitha and Annapurna and my extended family for their support. In particular, I would like to thank my uncle Sury Tumuluri, aunt Rohini Tumuluri and brother-in-law Rama Gundlapalli for showing me the vision that aided my decision to pursue a Ph.D. degree. My thanks also to my long-time friends and fellow graduate students Vinay Nadig, Karthik Venkataraman, Amresh Maadhavan, and Emad Felemban for going out with me on coffee-breaks. Last but definitely not the least, I thank my wife Sudha for all her love, humor and tireless efforts that gave me the energy to complete the work presented in this dissertation.
VITA

March 16th, 1978 .......................... Born - Bangalore, India

1995 - 1999 ............................... B.E. Electrical and Electronics Engg.,
Bangalore University
1999 - 2000 ............................... Associate Systems Engineer,
MphasiS, Bangalore, India
2000 - 2002 ............................... Graduate Research Associate,
The Ohio State University
2000 - 2002 ............................... M.S. Electrical and Computer Engg.,
The Ohio State University
2002 - 2005 ............................... Systems Developer/Engineer,
OARnet/Ohio Supercomputer Center
2005 - Present ............................ Senior Systems Developer/Engineer,
OARnet/Ohio Supercomputer Center

PUBLICATIONS

Research Publications

for Online VVoIP QoE Measurement”, Journal of Communications and Networks (JCN),
2007.

P. Calyam, M. Haffner, E. Ekici, C.-G. Lee, “Measuring Interaction QoE in Internet Video-
conferencing”, IEEE/IFIP Management of Multimedia and Mobile Networks and Services

P. Calyam, C.-G. Lee, E. Ekici, M. Haffner, N. Howes, “Orchestrating Network-wide Ac-
tive Measurements for Supporting Distributed Computing Applications”, IEEE Transac-

viii


FIELDS OF STUDY

Major Field: Electrical and Computer Engineering
TABLE OF CONTENTS

Abstract . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . ii
Dedication . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . iv
Acknowledgments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . v
Vita . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . viii
List of Figures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xiv
List of Tables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . xix

Chapters:

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1
   1.1 Motivation and Significance . . . . . . . . . . . . . . . . . . . . . . . . 1
      1.1.1 Internet Growth . . . . . . . . . . . . . . . . . . . . . . . . . . 1
      1.1.2 Internet Performance Measurements . . . . . . . . . . . . . . . 3
      1.1.3 Application-specific Measurement Issues . . . . . . . . . . . . . 6
   1.2 Approach . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
1.3 Dissertation Outline ........................................... 15

2. Background and Related Work ........................................ 17
   2.1 Network Quality of Service (QoS) Metrics ......................... 18
   2.2 On-going and On-demand Measurements ......................... 23
   2.3 Measurement Scheduling Challenges .......................... 28
      2.3.1 Measurement Conflict Avoidance ....................... 28
      2.3.2 Measurement Regulation ................................ 30
      2.3.3 Centralized and Distributed Schedule Management ..... 31
      2.3.4 Multi-domain Measurements ........................... 32
   2.4 Round-robin Scheduling Algorithms .......................... 34
   2.5 Voice and Video over IP (VVoIP) Quality of Experience (QoE) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
      2.5.1 VVoIP System Overview ................................ 36
      2.5.2 Streaming and Interaction VVoIP QoE ................. 38
   2.6 Network QoS and VVoIP QoE Relation ......................... 40
      2.6.1 Mean Opinion Score (MOS) Metric ...................... 40
      2.6.2 Subjective and Objective VVoIP QoE ................... 42
      2.6.3 VoIP QoE Estimation Techniques ...................... 43
      2.6.4 VVoIP QoE Estimation Techniques ...................... 45
   2.7 Summary ................................................... 47

3. Multi-domain Measurement Scheduling .......................... 49
   3.1 Concurrent Execution (CE) Principle ......................... 50

xi
4.1.6 GAP-Model Validation ........................................ 124
4.2 VVoIP Interaction QoE Estimation .......................... 128
  4.2.1 Low and High Video Activity Levels ...................... 129
  4.2.2 Network Fault Events ................................... 131
  4.2.3 Multi-Activity Packet Trains (MAPTs) .................... 132
  4.2.4 Video Packet Size Distribution Analysis ................. 142
  4.2.5 Video Encoding-Rate Time Series Analysis ............... 143
  4.2.6 MAPTs implementation in Vperf Tool ....................... 146
  4.2.7 MAPTs Evaluation ..................................... 148
4.3 Summary ..................................................... 152

5. Conclusions and Future Work .................................. 154
  5.1 Contributions ................................................ 154
  5.2 Open Issues and Future Directions ........................ 159

Bibliography ...................................................... 163
LIST OF FIGURES

Figure                                Page

2.1  Packet pair dispersion             22
2.2  Measurement topology in an NMI     24
2.3  Sampling patterns of on-going measurements 26
2.4  Scheduling active measurement tasks versus scheduling OS-threads and packets 29
2.5  Iperf experiment results with and without mutual exclusion of measurements 31
2.6  Centralized and distributed scheduling 32
2.7  Multi-Domain NMI showing end-to-end measurement paths 33
2.8  End-to-end view of a VVoIP system 36
2.9  A videoconference session showing video impairments 39
2.10 MOS rankings for Good, Acceptable and Poor grades of user satisfaction 41
2.11 Mapping between R-factor values and MOS rankings 44
2.12 Mapping of PSNR values to MOS rankings 46
3.1  Example task set to show task conflict graph construction 51
3.2  Example measurement schedule satisfying MLA $\psi$ 53
3.3 Heuristic bin packing flowchart ..................................... 54
3.4 Measurement topologies: (a) Full-mesh, (b) Tree, and (c) Hybrid ........ 55
3.5 Cycle time comparison for round-robin and heuristic bin packing ........ 56
3.6 Effects of increase in number of measurement tools on cycle times in heuristic bin packing ........................................... 56
3.7 Effects of bin sizes on cycle times in heuristic bin packing ............ 58
3.8 No orchestrated schedule ................................................... 59
3.9 Orchestration based on single processor non-preemptive EDF schedule ... 60
3.10 EDF-CE Schedule ......................................................... 61
3.11 Maximum schedulable utilization by three scheduling algorithms ........ 64
3.12 Effect of MLA $\psi$ and conflict factor to EDF-CE ..................... 65
3.13 Recursive pushing for maximum slack calculation ..................... 67
3.14 Average response time of on-demand jobs ................................ 71
3.15 Online schedule overhead for on-demand jobs .......................... 71
3.16 Minimal spanning tree for the measurement server topology .......... 72
3.17 Distributed schedule update ............................................. 73
3.18 Response time comparison of the centralized and distributed implementations 76
3.19 Implementation overhead comparison of the centralized and distributed implementations ......................................................... 77
3.20 Request/response schema example to exchange inter-domain measurements 79
3.21 ActiveMon framework components ..................................... 80
3.22 Structure of measurement scheduling framework ................. 81
3.23 Internet2 Testbed Setup ........................................... 82
3.24 H.323 Beacon MOS measurements between Site-3 and Site-4 .... 83
3.25 H.323 Beacon MOS measurements over a week period between Site-3 and Site-4 .............................................. 84
3.26 Iperf jitter measurements between Site-3 and Site-2 ............. 84
3.27 Pathload bandwidth measurements between Site-3 and Site-5 .... 85
3.28 Route changes detected along campus backbone network path .... 88
3.29 Delay anomalies detected along campus backbone network path .... 89
3.30 Jitter anomalies detected along campus backbone network path .... 90
3.31 OWAMP “ocean-wave anomaly” detected on regional backbone network path .............................................. 90
3.32 Bandwidth anomaly detected on regional backbone network path .... 91
3.33 Jitter of inter-sampling time points of network status .......... 94
3.34 APS transformation to TPS .......................................... 95
3.35 Four task example for determining inter-sampling times of APS data .... 96
3.36 Comparison of Forecasts of PPS and APS ......................... 97
3.37 Comparison of Forecasts of PPS and TPS .......................... 98
3.38 Comparison of MSE of PPS, APS and TPS ....................... 99
4.1 $j_{net}$ measurements for increasing $b_{net}$ .......................... 104
4.2 $l_{net}$ measurements for increasing $b_{net}$ .......................... 105
4.3 (a) VF test case ordering (b) HF test case ordering .......... 106

xvi
4.4 Test environment setup for the closed-network testing .................................. 108
4.5 Screenshot of chat application with quality assessment slider .......................... 108
4.6 Comparison of Streaming MOS (S-MOS) and Interactive MOS (I-MOS) .......... 116
4.7 Comparison of average S-MOS of Expert, General and Novice human subjects ................................................................. 117
4.8 Comparison of average, lower bound, and upper bound S-MOS ......................... 118
4.9 Training S-MOS response surface showing \( l_{\text{net}} \) and \( j_{\text{net}} \) effects .............. 121
4.10 GAP-Model S-MOS response surface showing \( l_{\text{net}} \) and \( j_{\text{net}} \) effects .......... 121
4.11 Training I-MOS response surface showing \( l_{\text{net}} \) and \( j_{\text{net}} \) effects .............. 121
4.12 GAP-Model I-MOS response surface showing \( l_{\text{net}} \) and \( j_{\text{net}} \) effects .......... 121
4.13 GAP-Model S-MOS response surface showing \( l_{\text{net}} \) and \( d_{\text{net}} \) effects .......... 122
4.14 GAP-Model S-MOS response surface showing \( l_{\text{net}} \) and \( b_{\text{net}} \) effects .......... 122
4.15 Online VVoIP QoE Measurement Framework ........................................... 123
4.16 Comparison of S-MOS with Validation-S-MOS (V-S-MOS) ......................... 126
4.17 Comparison of I-MOS with Validation-I-MOS (V-I-MOS) ............................. 126
4.18 Comparison of S-MOS with P-MOS .......................................................... 128
4.19 \( b_{\text{video}} \) for Claire (low \( a_{\text{lev}} \)) video sequence ........................................ 129
4.20 \( b_{\text{video}} \) for Foreman (high \( a_{\text{lev}} \)) video sequence ................................... 130
4.21 LH packet trains to emulate video activity levels for ‘listen’ followed by
‘talk’ end-user actions ......................................................................................... 133
4.22 General videoconference session agenda-items .......................................... 134

xvii
4.23 Videoconference session demand and supply for $PIP_1$ .......................... 136
4.24 Videoconference session demand and supply for $PIP_2$ .......................... 137
4.25 Videoconference session demand and supply for $PIP_3$ .......................... 137
4.26 Agenda-bandwidth for the three participant interaction patterns .............. 139
4.27 $b_{video}$ for combined low $a_{lev}$ video sequences ................................. 141
4.28 $b_{video}$ for combined high $a_{lev}$ video sequences ................................. 141
4.29 $tps$ distribution for high $a_{lev}$ at 256 Kbps dialing speed; $\alpha = 4.321$, $\beta = 110$ 142
4.30 (a) ACF of $m_t$ (b) ACF of $Y_t$ - for high $a_{lev}$ at 256 Kbps dialing speed ... 144
4.31 AIC for high $a_{lev}$ at 256 Kbps dialing speed ........................................ 145
4.32 Vperf tool components and their workflow ............................................. 147
4.33 An example Vperf tool session agenda specification .................................. 147
4.34 $LH$ packet trains emulation by Vperf at 256 Kbps dialing speed ............... 149
4.35 $LH$ packet trains emulation by Vperf at 384 Kbps dialing speed ............... 150
4.36 $LH$ packet trains emulation by Vperf at 768 Kbps dialing speed ............... 150
4.37 Impact of increasing number of Type-I network fault events on Agenda-bandwidth .......................................................... 151
4.38 Impact of increasing number of Type-II network fault events on Agenda-bandwidth .......................................................... 152
4.39 Impact of increasing number of Type-I and Type-II network fault events on Agenda-time at 768 Kbps dialing speed ........................................ 152
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Higher-end broadband connection technologies and their access speeds</td>
<td>2</td>
</tr>
<tr>
<td>2.1 $q_{mos}$ GAP grades and performance levels of network factors for $[b_{video}] = 768$ Kbps</td>
<td>42</td>
</tr>
<tr>
<td>3.1 ActiveMon measurement toolkit</td>
<td>81</td>
</tr>
<tr>
<td>3.2 Relative performance of the academic backbone network paths</td>
<td>92</td>
</tr>
<tr>
<td>4.1 Values of network factors within GAP performance levels for NISTnet configuration</td>
<td>114</td>
</tr>
<tr>
<td>4.2 Regression surface model parameters $C_0$ to $C_4$ for the six GAP-Model $q_{mos}$ types</td>
<td>119</td>
</tr>
<tr>
<td>4.3 Regression surface model parameters $C_5$ to $C_8$ for the six GAP-Model $q_{mos}$ types</td>
<td>119</td>
</tr>
<tr>
<td>4.4 Values of network factors for GAP-Model validation experiments</td>
<td>125</td>
</tr>
<tr>
<td>4.5 Gamma distribution parameters for low and high $a_{lev}$ video packet sizes</td>
<td>143</td>
</tr>
<tr>
<td>4.6 MA(2) parameters for low and high $a_{lev}$ video encoding rates</td>
<td>144</td>
</tr>
</tbody>
</table>
# LIST OF ALGORITHMS

1. **EDF-CE**: given a task conflict graph, find measurement schedule in a hyperperiod .................................................. 60
2. **push**: return the new start time of input jobs after maximum push ........ 68
3. Pseudo-code of the control software for streaming tests ..................... 110
4. Pseudo-code of the control software for interactive tests .................... 111
## GLOSSARY

<table>
<thead>
<tr>
<th>Term/Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACF</td>
<td>Auto-Correlation Function</td>
</tr>
<tr>
<td>APS</td>
<td>Average Periodic Sampling</td>
</tr>
<tr>
<td>CE</td>
<td>Concurrent Execution</td>
</tr>
<tr>
<td>DDoS</td>
<td>Distributed Denial of Service</td>
</tr>
<tr>
<td>EDF</td>
<td>Earliest Deadline First</td>
</tr>
<tr>
<td>GAP</td>
<td>Good, Acceptable and Poor</td>
</tr>
<tr>
<td>ISP</td>
<td>Internet Service Provider</td>
</tr>
<tr>
<td>ITU</td>
<td>International Telecommunications Union</td>
</tr>
<tr>
<td>MA</td>
<td>Moving Average</td>
</tr>
<tr>
<td>MAPTs</td>
<td>Multi-Activity Packet Trains</td>
</tr>
<tr>
<td>MCU</td>
<td>Multi-point Control Unit</td>
</tr>
<tr>
<td>MLA</td>
<td>Measurement Level Agreement</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>NMI</td>
<td>Network Measurement Infrastructure</td>
</tr>
<tr>
<td>NTP</td>
<td>Network Time Protocol</td>
</tr>
<tr>
<td>NWS</td>
<td>Network Weather Service</td>
</tr>
<tr>
<td>OWAMP</td>
<td>One-Way Active Measurement Protocol</td>
</tr>
<tr>
<td>PIP</td>
<td>Participant Interaction Pattern</td>
</tr>
<tr>
<td>PPS</td>
<td>Pure Periodic Sampling</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of Experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>TPS</td>
<td>Transformed Periodic Sampling</td>
</tr>
<tr>
<td>VoIP</td>
<td>Voice over IP</td>
</tr>
<tr>
<td>VVoIP</td>
<td>Voice and Video over IP</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Motivation and Significance

1.1.1 Internet Growth

The beginnings of the Internet can be traced back to early 1960s. Since then, the Internet has grown at an unconceivable rate and is becoming almost ubiquitous. The success of the Internet can be gauged by the ever increasing number of end-hosts, end-users and world-wide geographical locations where Internet is being made accessible. The Internet was initially used primarily for data communication applications such as E-mail and FTP. However, in the last decade, there has been an exponential growth of advanced applications that involve voice, video and data communications over the Internet. These advanced applications use network protocols which are both complex and resource intensive and include: e-business, chat, music and photo sharing, video streaming, videoconferencing, Grid computing, IPTV, remote instrumentation, disaster recovery backups and online gaming. Even traditional desktop applications such as calendars, spreadsheets and word processors have become Internet-dependent. Further, mobile devices that traditionally were used for voice communications over wireless networks are evolving into Internet appliances that support
all the advanced applications. Hence, the Internet has become an integral part of our current society and is changing the traditional ways of research, business, communication and entertainment.

**Broadband Internet Connections**

Using the advanced applications requires high-speed access to the Internet or “broadband connections” at the end-hosts. The availability of broadband connections has become a necessity in the society today. It is being touted as an important consumer utility similar to electricity and water, and as a major driver for economic development. There are a couple of widely-accepted definitions of broadband connections. The U.S. Federal Communications Commission (FCC) defines broadband connection downstream data rates to be 256 Kbps and higher, whereas, the International Telecommunications Union (ITU) defines the same to be 1.5 Mbps and higher. Nevertheless, higher-end broadband connections are being provided today to businesses by Internet Service Providers (ISPs) at extremely high data-rates using technologies such as the ones shown in Table 1.1.1 [1].

<table>
<thead>
<tr>
<th>Technology</th>
<th>Connection Name</th>
<th>Connection Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDM Circuit-based</td>
<td>DS3</td>
<td>45 Mbps</td>
</tr>
<tr>
<td>TDM Circuit-based</td>
<td>OC3</td>
<td>155 Mbps</td>
</tr>
<tr>
<td>TDM Circuit-based</td>
<td>OC12</td>
<td>622 Mbps</td>
</tr>
<tr>
<td>Ethernet-based</td>
<td>Ethernet</td>
<td>10 Mbps</td>
</tr>
<tr>
<td>Ethernet-based</td>
<td>Fast Ethernet</td>
<td>100 Mbps</td>
</tr>
<tr>
<td>Ethernet-based</td>
<td>Gigabit Ethernet</td>
<td>1000 Mbps</td>
</tr>
</tbody>
</table>

Table 1.1: Higher-end broadband connection technologies and their access speeds
To interconnect such broadband connections of end-hosts on a global-scale, ISPs are building massive backbone networks and interconnecting them with sophisticated routing protocols at the regional, national and inter-national levels. Using such hierarchical network architectures, ISPs are able to carry voice, video and data traffic with data rates on the order of several hundreds of Gbps on the Internet.

1.1.2 Internet Performance Measurements

To successfully maintain their networks that handle such large volumes of voice, video and data traffic, ISPs need to address several challenges. One challenge is to maintain availability of their networks. Any network downtimes are instantly noticeable to end-users, who will then complain. For example, if an end-user is downloading a file, a network downtime will cause the download to stall abruptly. Another challenge is to maintain reliability of their networks. If there are any network fault events caused due to reasons that include: (a) cross-traffic congestion at intermediate hops, (b) physical link fractures, and (c) Distributed Denial of Service (DDoS) attacks, application performance will be impacted that will degrade end-user satisfaction. For example, if an end-user is in a videoconference session, the network fault events will manifest to end-users as perceptual quality impairments such as video frame freezing and voice dropouts. Yet another challenge arises due to the hierarchical structure of the Internet. Given that ISPs need to depend on each other to carry each other’s end-user traffic to any destination on the Internet, they have to meet the availability and reliability expectations of neighboring ISP’s end-users, as well. If network downtimes and fault events occur frequently, ISPs run into the risk of losing valuable business from their end-users and neighboring ISPs in today’s competitive Internet services market.
For setting up the network availability and reliability expectations of their end-users and neighboring ISPs, ISPs engage in and adhere to Service Level Agreements (SLAs). The SLAs are essentially legal contracts that contain mutually agreeable levels of network availability and Quality of Service (QoS) metrics that relate to network performance reliability. The QoS metrics include access bandwidth, packet delay, jitter and loss. It is known that different levels of these QoS metrics directly correlate to application performance and ultimately the end-user Quality of Experience (QoE) grades [2] [3]. For these reasons, ISPs constantly strive to maintain high availability of their networks and regularly monitor their networks to ensure the QoS metrics are in good performance-levels.

To routinely monitor network performance and troubleshoot network bottlenecks that impact SLA compliance, ISPs have started instrumenting their networks with Network Measurement Infrastructures (NMIs). The NMIs allow provisioning and analysis of QoS metrics along strategic network paths. Examples of NMIs include NLANR AMP [4], Flowtools [5], RIPE [6] and Internet2 Abilene Observatory [7]. For provisioning the QoS metrics, NMIs use active and passive network measurement tools. Active measurement tools inject a series of probing packets into the network to measure the QoS metrics. Popular active measurement tools include Ping, Traceroute, OWAMP [8], Pathrate [9], Pathchar [10] and Iperf [11]. In comparison, passive measurement tools do not inject probing packets into the network. They capture application-traffic’s packet-level information such as packet numbers, bytes, CRC errors, and timestamps at network switches and routers to measure the QoS metrics. Popular passive measurement tools include TCPdump [12], NetFlow [5]

The primary focus of the research presented in this dissertation relates to provisioning and analysis approaches relevant to active measurements on the Internet. However, our approaches can also be applied to passive measurements with suitable modifications.
and Simple Network Management Protocol (SNMP) [13]. Thus provisioned measurements are analyzed using various techniques. Commonly, ISPs use time-series plots, summary statistics and simple threshold-based performance anomaly detection techniques for routine monitoring of network paths. In addition, ISPs also resort to automated frameworks such as Network Weather Service (NWS) [14], E-Model [15] and Snort [16] that use sophisticated estimation techniques for analysis of network path performance. NWS is used to forecast network performance trends, E-Model is used to estimate end-user QoE of VoIP applications, and Snort is used to detect network intrusions such as DDoS attacks.

**Multi-domain Measurements**

Deploying an NMI generally helps an ISP to isolate network bottlenecks within the ISP’s domain. However, troubleshooting network bottlenecks involving other ISP’s domains requires access to Internet measurements collected by the other ISP’s NMI. The process of facilitating such an exchange of Internet measurements between ISPs requires dealing with complex privacy and policy issues rather than technical issues. Nevertheless, ISPs in both academia and industry have started publicly disclosing their Internet measurement data sets of their NMI-domains for deriving mutual benefits [4] [7] [17] [18] [19]. There are even “Request/Response” schemas being standardized by the Global Grid Forum’s Network Measurements Working Group (NMWG) [20] that specify formats for Internet measurements data exchange in an automated manner between multi-domain NMIs. Further, through community forums such as [21], organizations such as Internet2 in USA and DANTE in Europe are forming multi-domain NMI federations involving national and international ISPs.
1.1.3 Application-specific Measurement Issues

In spite of the above efforts of ISPs towards SLA-compliance, last-hop bottlenecks and the sheer complexity/scale of the Internet cause the end-to-end QoS metrics of network paths between any two end-hosts to intermittently fall into poor performance-levels [22] [23]. Such instances particularly cause performance problems i.e., they impact end-user QoE in applications that rely on computing resources distributed across the Internet. For example, Grid computing end-users often transfer computational jobs involving large data sets (sometimes even on the scale of terabytes) using file transfer applications to remote supercomputing sites. If such jobs traverse problematic network paths with excessive packet delays and loss, they could hold up the speedy completion of other queued jobs that traverse problem-free network paths, thus increasing the wait-time of end-users waiting for their job completions. As another example, let us consider a multi-party videoconferencing application where three or more participants are interested in having an interactive session. For this, they connect to a call admission controller that chooses one of the several Multi-point Control Units (MCUs) it manages. An MCU combines the admitted voice and video streams from participants and generates a single conference stream that is multicast to all the participants. If a call-admission controller selects a problematic network path with excessive packet jitter between the MCU and the participants, the perceptual quality of the conference stream could be seriously affected by impairments such as video-frame freezing and voice drop-outs, and even call-disconnects. Such impairments obviously impact the perceptual and interaction QoE of the participants. Thus, from the two application examples, we can see that each application has unique measurement issues that need to be addressed by the frameworks for identifying network bottlenecks and for performing resource adaptation to obtain optimum end-user QoE.
Application Adaptation

Managing the selection of network paths between networked computing resources and end-users to ensure satisfactory end-user QoE is generally beyond the realm and expertise of ISPs. Hence, ISPs collaborate with Application Service Providers (ASPs) who rely on the frameworks that can control and manage the networked computing resources for end-users. Typically, for redundancy and load-balancing reasons, ASPs use overlay networks and multi-homing that requires subscription to multiple ISPs to deliver their services. MPLS explicit routing and techniques such as [24] have been developed that exploit path diversity based on multi-homing or overlay networks. Using these techniques, the frameworks can be configured to dynamically select the appropriate network paths between the end-user and the networked computing resources. Also, the frameworks can employ bandwidth-on-demand techniques [25] on network paths in situations where network path selections are primarily affected by last-hop access bandwidth limitations.

To have the ability to dynamically determine the most appropriate network path amongst multiple network path options or employ bandwidth-on-demand, the frameworks require detailed network status information. Specifically, they need suitable provisioning and subsequent real-time analysis of QoS metrics pertaining to several network paths. The provisioning has to facilitate as frequent as possible sampling of QoS metrics on all the monitored network paths. This can enable rapid detection and appropriate adaptation based on historical as well as latest active measurements data to circumvent impending network performance problems that seriously affect end-user QoE. In addition, if the QoS metrics need to be analyzed for accurate network weather forecasting using frameworks such as NWS, the measurement sampling needs to be strictly periodic [14].
Voice and Video over IP Application Issues

Even if the measurements are provisioned as desired, frameworks supporting networked computing applications involving Voice and Video over IP (VVoIP) still face unique analysis challenges. This is because end-users of VVoIP applications such as videoconferencing are extremely sensitive to voice and video impairments caused by intermittent network fault events compared to say the Grid computing end-users, whose jobs may take a few seconds more to complete under similar conditions. Although VVoIP end-points today have dejitter buffers and packet loss concealment schemes to cope with the intermittent network fault events, it is not always possible for these schemes to mitigate the impairment effects, particularly those relating to lack of lip-synchronization [26] [27]. Hence, the frameworks need to employ estimation techniques that can fittingly map the network QoS levels to Good, Acceptable and Poor end-user QoE grades in real-time. Such estimation techniques can also help in avoiding the process of involving end-users, which is: (a) expensive, (b) time consuming, and (c) a hindrance for framework automation.

Although there have been prior efforts on developing techniques to provision and analyze Internet measurements that address the aforementioned application-specific measurement issues in frameworks, there are several key questions that have not been answered. The questions include:

- Which scheduling algorithms can optimally provision network-wide measurements with any desired frequency and periodicity?

- How can measurements be regulated and provisioned between multiple ISP domains?

- What kinds of network performance anomalies and trends can be detected from provisioned measurements over multi-resolution time scales on the Internet?
• How can we analyze the network performance measurements to estimate end-user QoE for applications particularly involving VVoIP?

The lack of pertinent answers to these questions is limiting the degree of success achievable by the frameworks used to deliver optimum end-user QoE. Such a limitation threatens the wide-adoption and prevalence of existing and emerging VVoIP and and other real-time applications on the Internet that have a significant impact on our society and economy.

Dissertation Focus

Based on the above motivations and discussion, the focus of the research presented in this dissertation is as follows: To enable the provisioning and analysis of network QoS measurements as required by frameworks that manage resources to ensure optimum end-user QoE of advanced Internet applications such as videoconferencing and Grid computing.

1.2 Approach

Measurement Requirements and Constraints

For provisioning network-wide QoS metrics to the frameworks, we first studied the kinds of active measurement requests submitted to NMIs. Broadly, the requests can be grouped into either on-going or on-demand measurements. On-going measurements involve continuous sampling of network status at pre-specified time intervals. Such measurements are used for monitoring and forecasting network path performance. In comparison, on-demand measurements are one-off requests that are used to quickly collect customized measurements for performance troubleshooting. Next, we studied the constraints that need to be addressed for scheduling measurements to handle the on-going and on-demand measurement requests. A major constraint we discovered is the “measurement
conflict problem”. Since most active measurement tools inject probing packets and consume non-negligible amount of network resources, concurrently running two sets of active measurements on overlapping network segments can cause measurement conflicts that produce inaccurate reports of QoS metrics. Also, given the resource-intensive nature of active measurement tools, active measurements need to be regulated so that they do not overly consume network resources needed by actual application traffic.

**Measurement Scheduling Algorithms**

After studying the measurement requirements and constraints, we performed a literature survey of the existing scheduling algorithms being used in NMIs. We found that the existing techniques are based on the round-robin scheduling principle. As a result, only one measurement is executed at a time in the NMI, which severely limits the schedulability of measurement requests and affects frequent sampling of network status. To overcome this limitation, we propose a “concurrent execution” (CE) principle which allows concurrent execution or overlapping of non-conflicting measurements in the schedules if:

(a) measurements involve tools that are neither CPU and/or channel intensive, and

(b) measurements involve network paths that do not have common links.

Using the CE concept and borrowing real-time system scheduling principles such as heuristic bin packing and Earliest Deadline First (EDF) [28], we notably improve the schedulability of on-going measurements in NMIs. To improve the response time of on-demand measurements, we leverage online slack-determination principles used in real-time systems and propose a novel recursive push algorithm. This algorithm recursively pushes prescheduled measurements without violating any measurement requirements or constraints. It thus obtains maximum slack at the beginning of the measurement schedule within which
the on-demand measurement can be accommodated with minimal response time. To regulate the measurements, we propose using a “Measurement Level Agreement” (MLA) that specifies the percentage (e.g. 1% - 5%) or bandwidth consumption levels (e.g. 1 Mbps - 2 Mbps) of the measurement traffic. The MLA constraint is considered by our scheduling algorithms for limiting the number of CE measurements on a network path.

**Distributed Measurement Scheduling**

While initially developing our scheduling algorithms, we assumed a centralized implementation where a central regulator collects the on-going and on-demand measurement requests and processes them to build the network-wide measurement schedules. To avoid incapacitating an NMI when the central regulator fails, we propose a mechanism to implement our scheduling algorithms in a distributed manner. Another motivation for distributed schedule management is to enable applications that require flexibility in dynamically determining the locations of measurement data collection and analysis. Our distributed implementation of our scheduling algorithms uses the Raymond’s algorithm [29] that has been proven to possess properties such as minimal message exchanges, deadlock-freedom, no-starvation and fault-tolerance.

**Multi-domain Measurement Scheduling**

Another assumption while initially developing our scheduling algorithms was that our scheduling algorithms would be used intra-domain i.e., they will be used within a single ISP’s domain. However, in reality, measurements need to be collected across multiple ISP domains. For example, ASPs such as Vonage rely on NMIs of multiple ISPs domains for monitoring performance and delivering world-wide VoIP services. For facilitating the inter-domain measurements, we envision and describe how our measurement scheduling
algorithms can be used in “Active Measurement Federations” involving several ISPs who mutually agree upon sharing each other’s NMI information such as: (a) instrumented network paths, (b) MLAs, and (c) authentication for secure access to measurement resources. We also identify such federations and related measurement resource sharing standards that are emerging in communities such as the Global Grid Forum, Internet2 in USA and DANTE in Europe.

**Measurement Scheduling Case Studies**

To assess the utility of our scheduling algorithms, we implement our measurement scheduling algorithms in the ActiveMon [30] framework, which we developed. ActiveMon is useful for routine network-wide monitoring using active measurements. By virtue of our implementation in ActiveMon, we verify the trustworthiness of our scheduling algorithms to avoid measurement conflicts. Further, we deploy ActiveMon on a network testbed comprising of academic backbone network paths at the campus, regional and national levels. From our deployment, we:

(a) empirically correlate network events and notable performance trends,

(b) study characteristics of short-term and long-term network performance anomalies, and

(c) assess relative performance of the network paths.

We also implement our algorithms in the widely-used Network Weather Service (NWS) [14] to assess the utility of our scheduling algorithms for network weather forecasting. From our NWS implementation experience, we realize that no measurement scheduling algorithm that considers the measurement conflict constraint can cater to the strict periodicity requirements required for accurate network weather forecasting. However, we demonstrate
that this limitation can be overcome by using a piece-wise linear interpolation technique that transforms the measurements data in a manner that produces relatively more accurate network weather forecasts.

Measurement Analysis for VVoIP Applications

We next study techniques that analyze the provisioned network-wide QoS metrics in frameworks that support VVoIP applications such as videoconferencing and video-on-demand. It is known that voice and video quality degradations due to network dynamics manifest to end-users differently and ultimately impact their QoE. For example, voice quality degradations manifest in the form of impairments such as voice dropouts and echoes, whereas, video quality degradations manifest in the form of impairments such as video frame freezing and tiling [31]. It is not practical to rely on actual end-users to provide their subjective QoE assessments that indicate voice/video quality degradation when monitoring network paths. Hence, VVoIP supporting frameworks use objective techniques such as ITU-T E-Model [15] and the ITU-T J.144 [32] that estimate end-user QoE for any given network condition. These techniques analyze the network QoS metrics and employ psychoacoustic/visual cognitive models built using extensive offline human subject studies. The E-Model estimates end-user QoE only for VoIP applications and is not suitable for estimating end-user VVoIP QoE. Although the J.144 estimates end-user VVoIP QoE, it involves offline processing of sender-side and receiver-side video sequences to obtain Peak Signal-to-Noise Ratio (PSNR) measurements. Such a process is extremely time consuming and computationally intensive. Hence it is not useful for online VVoIP QoE monitoring, which is vital for dynamic adaptation of resources to maintain optimum end-user QoE.
Online VVoIP QoE Estimation

To address the lack of online VVoIP performance monitoring techniques, we develop a novel psycho-acoustic/visual cognitive model called “GAP-Model” that can provide online objective QoE estimates without end-user involvement and without using actual video sequences. To develop the GAP-Model, we map the combined effects of QoS metrics such as bandwidth, delay, jitter and loss to Good, Acceptable and Poor (GAP) grades of end-user QoE. We obtain the GAP grades from human subjects for different QoS metric levels using the widely-used “Mean Opinion Score” (MOS) ranking method [2]. This method comprises of a subjective quality scale of 1 to 5 - the [1, 3) range corresponds to Poor grade, [3, 4) range corresponds to Acceptable grade and [4, 5] range corresponds to Good grade. Thus, if statistically stable measures of QoS metrics are provisioned, the GAP-Model’s closed-form expressions instantly produce GAP grades of VVoIP QoE. We validate the GAP-Model’s QoE estimates for both streaming VVoIP (e.g. video-on-demand) and interactive VVoIP (e.g. videoconferencing) applications. Finally, we propose a novel active measurement methodology to quantify VVoIP interaction QoE in terms of the interaction difficulties experienced by end-users on a network path. The methodology involves Multi-Activity Packet Trains (MAPTs) that mimic a videoconference session’s participant interaction patterns and the corresponding video activity levels affected by intermittent network fault events. To emulate MAPTs, we develop empirical models for characterizing salient VVoIP traffic behavior in terms of video encoding rate and packet size distribution. We implement both our GAP-Model and MAPTs in a tool we developed called Vperf that can be scheduled in NMIs using our measurement scheduling algorithms for online VVoIP QoE estimation.
1.3 Dissertation Outline

In Chapter 2, we present the technical background and review related work. In Section 2.1, we define the network QoS metrics and describe the tools used to measure them. In Section 2.2, we describe the on-going and on-demand measurement requests submitted to provision measurement data. In Section 2.3, we describe the various measurement scheduling challenges. Specifically, in Section 2.3.1, we illustrate the measurement conflict problem. In Section 2.3.2, we describe the measurement regulation constraint. In Section 2.3.3, we compare and contrast centralized and distributed schedule management. In Section 2.3.4, we explain the technical and policy issues involved in provisioning multi-domain measurement. In Section 2.4, we describe the existing measurement scheduling algorithms. In Section 2.5, we introduce the importance of end-user QoE in a VVoIP system. First, in Section 2.5.1, we study the factors that impact end-user QoE. Following this, in Section 2.5.2, we explain the two different end-user VVoIP QoE types i.e., streaming and interactive. In Section 2.6, we describe the inter-relation between network QoS and VVoIP QoE. In Section 2.6.1, we introduce the Mean Opinion Score (MOS) metric that can be used to quantify end-user QoE. In Section 2.6.2, we differentiate the subjective and objective VVoIP QoE. In Sections 2.5.3 and 2.5.4, we discuss existing techniques such as the E-Model and J.144 estimation standardized by ITU-T to provide objective estimates of VoIP and VVoIP QoE, respectively, without actual end-user involvement.

In Chapter 3, we explain our measurement scheduling algorithms that cater to the measurement requirements without violating the scheduling constraints. In Section 3.1, we present our Concurrent Execution (CE) principle to improve schedulability. In Section 3.2, we define our measurement Level Agreement (MLA) parameter that enforces measurement
regulation on a network path. In Section 3.3, we explain our measurement scheduling algorithms. In particular, in Section 3.3.1, we explain our heuristic bin packing scheduling algorithm that improves schedulability but does not cater to any periodicity requirements. In Section 3.3.2, we explain our EDF-CE scheduling algorithm that improves measurement schedulability and also addresses periodicity requirements. In Section 3.3.3, we explain our recursive-push scheduling algorithm that schedules on-demand measurements with minimum response time. In Section 3.4, we describe a distributed implementation of our scheduling algorithms. In Section 3.5, we describe active measurement federations for provisioning multi-domain measurements. In Sections 3.6 and 3.7, we present case studies that show study the utility of our measurement scheduling algorithms.

In Chapter 4, we describe our Vperf tool for measuring streaming and interaction VVoIP QoE over the Internet. In Section 4.1, we present our GAP-Model. Specifically, in Sections 4.1.1 and 4.1.2, we explain our methodology to derive the GAP-Model. The GAP-Model’s closed-form expressions are explained in Section 4.2.3. In Section 4.1.4, we describe the GAP-Model characteristics. In Section 4.1.5, we describe how we implement the GAP-Model in the Vperf tool. In Section 4.1.6, we validate the GAP-Model with additional human subject tests and compare its estimates with the J.144 estimates. Then, in Section 4.2, we describe the MAPTs methodology. In particular, in Sections 4.2.1, 4.2.2 and 4.2.3, we explain the video activity levels involved in participant interaction patterns. In Sections 4.2.4 and 4.2.5, we explain the packet-train characteristics for emulating the MAPTs. In Section 4.2.6, we describe our MAPTs implementation in the Vperf tool. In Section 4.2.7, we evaluate the MAPTs methodology.

In Chapter 5, we list all the salient contributions of the dissertation. We also discuss several open issues and future research directions pertaining to our work.
CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we present the technical background and related work required to understand the in-depth technical content and dissertation contributions presented in the following chapters. Specifically, in Section 2.1, we define the different network QoS metrics that can be measured on the Internet using active measurements. We also describe various active probing techniques that have been implemented in measurement tools of NMIs to measure online network status. Next, in Section 2.2, we explain scenarios where automated frameworks request NMIs to provision network QoS information i.e., network status, on an on-going and on-demand basis. We also briefly explain how the provisioned data is used within the frameworks for network control and management to improve end-user QoE. In Section 2.3, we explain the measurement scheduling challenges that need to be addressed in NMIs to provision measurement data as needed by the frameworks. The challenges include: (a) measurement conflict avoidance, (b) measurement regulation, (c) centralized and distributed measurement schedule management, and (d) provisioning multi-domain measurements.

In Section 2.4, we describe existing round-robin measurement scheduling algorithms that attempt to cater to the scheduling challenges. In Section 2.5, we focus on the metrics and analysis techniques that estimate end-user QoE of VVoIP applications based on the
provisioned network QoS measurements. We start off with describing the human, device and network factors that impact end-user VVoIP QoE. Following this, we explain the significance of the two end-user VVoIP QoE types i.e., streaming and interactive. Lastly, in Section 2.6, we present the metrics that have been proposed in frameworks to map the interrelation between network QoS and end-user VVoIP QoE. In this context, we also examine techniques such as the E-Model and J.144 standardized by ITU-T to estimate end-user VoIP and VVoIP QoE, respectively.

2.1 Network Quality of Service (QoS) Metrics

Network QoS is quantified using performance metrics such as bandwidth, delay, loss and jitter. Bandwidth characterizes the amount of data that can be transferred on a network path per unit time. There are two common variants of this metric: bandwidth capacity and available bandwidth [33]. The bandwidth capacity refers to the maximum possible bandwidth that a link or a path can deliver. It is generally fixed and is dictated by the physical connection technology used (TDM circuit-based or Ethernet-based) as explained in Chapter 1. In contrast, available bandwidth refers to the maximum unused bandwidth in a link or a path. It varies due to the application traffic dynamics and occurrence of network fault events. Delay, also referred to as latency, is the time taken for a packet to traverse from a sender side to a receiver side. Commonly, “round trip time” (RTT) is used to characterize delay on a network path. Loss indicates the percentage of packets lost as observed at the receiver side for a given number of packets transmitted from the sender side. Finally, jitter is used to characterize the variations in the one-way network delays as seen at the receiver side.
ICMP-based Network Measurement Tools

There are several techniques that have been implemented within tools used in NMIs for online measurements of network QoS. The Internet Control Message Protocol (ICMP) is the core protocol used in online measurements. ICMP probing packets are sent from the sender side with a time-to-live (TTL) number that specifies the last hop at which the probing packets expire. Each intermediate hop decrements the TTL by one and once the TTL is set to zero at the last hop, a ‘TTL exceeded in transit’ response is sent back by the last hop to the sender side.

The popular Ping tool implementation uses ICMP “echo request” and “echo response” messages to determine delay and reachability between the sender and receiver sides. Specifically, by subtracting the timestamps of the echo request packet and the echo response packet, the round-trip delay to a particular receiver side is calculated. Also, by setting the TTL value to $n$, we can find the address of the $n^{th}$ router in the path. If an echo response is not received from an intended receiver side hop, the receiver side is deemed unreachable. Further, packet loss is calculated in Ping by sending multiple echo requests and checking whether responses are received for all the requests. The percentage of request packets that did not result in a response indicates the packet loss on the network path. To measure one-way delays on a network path, tools that implement the One-Way Active Measurement Protocol (OWAMP) [8] have been developed. OWAMP-based tools produce high-precision one-way delay and one-way loss using timestamps of UDP packets obtained from good time sources such as GPS and CDMA. It is relevant to note that the Network Time Control (NTP) protocol is used for time synchronization of all the OWAMP end-points.
Tools such as *Traceroute*, *Pathchar* [10] and *Pchar* [34] use an ICMP implementation similar to the one used in Ping. However, they focus on determining the “route” i.e., list of intermediate hops between sender side and receiver side. They also determine per-hop delay and per-hop available bandwidth. The per-hop delay is calculated by increasing the TTL of each successive batch of ICMP request packets towards the receiver side. *Pathchar* and *Pchar* estimate the per-hop available bandwidth by using the variable packet size (VPS) probing technique. This technique involves sending a series of ICMP probes with varying values of hops $n$ and packet sizes. The probes are single-ended in the sense that there is no need for any measurement software to be running at the receiver side. From these probes, RTT is measured from the source to each hop of the path as a function of the probing packet size. The slope of the linear interpolation of the minimum RTT measurements is treated as the inverse of the available bandwidth estimate at that hop.

**Available Bandwidth Estimation**

Besides the VPS probing technique, there are other packet probing techniques developed for end-to-end available bandwidth estimation. Amongst them, using Self-Loading Periodic Streams (SLoPS) is a popular technique. SLoPS are used in tools such as *Pathload* [9] and *Pathchirp* [35]. The source sends a number of equal-sized packets (e.g. a periodic packet stream of 100 packets) to the receiver at a certain rate $R$. The variations in the one-way delays of the probing packets are monitored. If the stream rate $R$ is greater than the path’s available bandwidth $A$, the stream will cause a short-term overload in the queue of a bottleneck link in the path. One-way delays of the probing packets will keep increasing as each packet of the stream queues up at the bottleneck link. On the other hand, if the stream rate $R$ is lower than the available bandwidth $A$, the probing packets will go through
the path without causing increasing backlog at the bottleneck link, and their one-way delays will not increase. While the sender probes with successive packet trains of different rates using a binary search as mentioned above, the receiver notifies the sender about the one-way delay trend of each stream. Using the trends in the changes of one-way delays, available bandwidth is estimated. Given the above methodology, SLoPS requires double-ended measurements i.e., measurement software needs to be running at both the source and the receiver sides of the path.

**Bandwidth Capacity Estimation**

To estimate the end-to-end bandwidth capacity, packet pair/train dispersion (PPTD) probing techniques are used. PPTD probing is used in tools such as Bprobe [36], Net.timer [37], which rely on packet pair dispersions. In tools such as Pathrate [9] and Cprobe [36], dispersions of packet trains are used. In PPTD, the source sends multiple packet pairs or packet trains to the receiver. Each packet pair consists of two packets of the same size, $L$ sent back to back as shown in Figure 2.1. We can notice the dispersion ($\Delta_{\text{out}}$) of the packet pair due to a router of a network path connected to a link with capacity $C_i$. After the packet pair reaches the receiver after traversing several links, $H$ in the network path, the receiver will measure dispersion $\Delta_{R}$. Thus, the receiver can estimate the path bandwidth capacity using the equation $C = L/\Delta_{R}$. Given the above probing methodology, PPTD probing also requires double-ended measurements.

In addition to the above explained variants of packet probing popularly used for bandwidth estimation, there are several other techniques that have been developed [38, 39]. Of the other techniques, a notable probing technique can be found in tools such as Iperf [11] and Netperf [40]. They use path flooding to determine achievable TCP or UDP throughput based on the available bandwidth on network paths. They allow controlling socket buffer
sizes and thus the maximum window size for TCP transfers. Given the above probing methodology, probing using path flooding also requires double-ended measurements.

VoIP QoE Estimation

For measuring performance of real-time applications involving VVoIP, instantaneous measurements of delay, jitter and loss are critical. Measurement tools such as the H.323 Beacon [41] use Real-time Transport Protocol (RTP) packets, which are essentially UDP protocol packets with special packet formatting. RTP uses a control protocol called RTP Control Protocol (RTCP) to send periodic network performance reports collected at the receiver side to the sender side of the H.323 Beacon. These reports contain information such as the number of lost packets since last report and estimated inter-arrival jitter of the packets in the RTP session. By plugging this information into the E-Model [15] (explained in detail in Section 2.6 of this chapter), the H.323 Beacon estimates the VoIP QoE on a network path.

Figure 2.1: Packet pair dispersion
2.2 On-going and On-demand Measurements

We now describe the on-going and on-demand measurement requests submitted by frameworks to NMIs. Processing the requests involves running the active measurement tools and provisioning network QoS measurements. Figure 2.2 shows an NMI deployed over the Internet2 Abilene network backbone. Measurement servers are deployed at strategic points and can be used to measure network paths to other servers. Further, the measurement servers are attached to core routers as shown in the case of the Denver core router. The paths to be measured are specified by a measurement topology, which can be formally represented by a graph $G = (N, E)$, where $N$ is the set of measurement servers and $E$ is the set of edges between a pair of servers. Figure 2 shows an example measurement topology that consists of measurement servers $N = \{S_1, S_2, S_3, S_4, S_5\}$ and edges among $S_1, S_2, S_3, S_4, S_5$.

On-going Measurements

On top of the measurement topology, a set of on-going measurement requests are specified. Recall from Chapter 1 that on-going measurements involve continuous sampling of network status at pre-specified time intervals. The network status information is used for real-time decision making for path selection (e.g. dynamic path switching) or resource allocation (e.g. bandwidth-on-demand, VVoIP call-admission control, codec type and bitrate selection) or even timely action invocation (e.g., DDoS attack-traceback upon detection). Using real-time systems terminology, an on-going measurement request can be treated as an “offline task” $\tau_i$. The $\tau_i$ specification contains the source server $src_i$, destination server $dst_i$, active measurement tool $tool_i$, inter-sampling time between consecutive measurement instances i.e., period $(p_i)$ and finally the execution time $e_i$ of a single measurement instance.
Thus, the task $\tau_i$ can be expressed as follows:

$$\tau_i = (src_i, dst_i, tool_i, p_i, c_i).$$

The set of all offline specified measurement tasks is denoted by:

$$\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}.$$ 

We define a hyperperiod for task set $\Gamma$ as the least common multiple of all the task periods in the set. In addition, we denote the $j$-th instance (or job) of $\tau_i$ as $\tau_{ij}$. Further, the time when the $j$-th job $\tau_{ij}$ is released is called the release time and simply given by $(j - 1)p_i$. It is relevant to note that offline tasks could be specified without the period $p_i$ information. Such tasks refer to measurements that do not have strict deadlines for each of the individual measurement jobs. They are commonly used in routine network monitoring.
scenarios where the goal is to minimize the overall measurements schedule time or cycle time. By minimizing the cycle time, measurements are initiated as frequently as possible for obtaining up to date network-wide status snapshots.

**Network Status Sampling**

In addition to demanding higher frequency of measurements, frameworks such as [42] - [49] demand higher flexibility of measurements for use by the analysis techniques i.e., they also require NMIs to meet strict deadlines of individual measurement jobs of corresponding tasks. Figure 2.3 shows the different inter-sampling time patterns of offline measurements that can be requested. The specific pattern requested depends on the monitoring accuracy objective, which is affected by the variance of the estimate of the actual network status. Lower the expected variance of the estimate, the more suited is the sampling method. *Pure periodic* sampling (also known as systematic sampling) shown in Figure 2.3(a) involves deterministically sampling network status at fixed and evenly-spaced time intervals. *Average periodic* sampling (also known as Stratified random sampling) shown in Figure 2.3(b) combines the fixed time-interval used in periodic sampling with random sampling by taking a single sample at a random point during a given time interval. *Random* sampling shown in Figure 2.3(c) employs a random distribution function (e.g. poisson, exponential) to determine when each network status sample should be taken. Pure periodic sampling is preferred by the network weather forecasting frameworks such as NWS [14], where the monitoring accuracy is known to be higher when there is an even spread in the sample population of measurements data. Average periodic and random sampling are preferred over periodic sampling by frameworks where there exists a positive correlation between pairs of samples in the population of measurements data. Using periodic sampling in above cases is known to produce higher variance in the estimation precision of the network status characteristics.
Further, average periodic sampling is known to be suitable when sample population have a linear trend [50].

In addition to the above three inter-sampling time patterns, there are frameworks such as [51] - [54] that use *adaptive* sampling. This is because adaptive sampling reduces the amount of processed data while still being able to accurately characterize network status. By using adaptive sampling, the sampling frequency can be increased or decreased based on assumptions such as diurnal patterns observed on networks. The diurnal patterns can be those that are mainly influenced by human behavioral patterns of using the Internet [55] [56] i.e., the network status changes based on the time-of-the-day/day-of-the-week or during times of events that affect human activity (e.g. increased traffic to News websites on occurrence of calamities). Figure 2.3(d) shows an example of adaptive sampling, where there is a higher frequency of sampling during the busy day time business hours and a relatively lower frequency of sampling during the night, when most businesses are closed. In general, adaptive sampling techniques inherently use the above three inter-sampling patterns within different frequencies of sampling over time. Hence, serving adaptive sampling
requests involves NMIs to provision pure periodic, average periodic and random samples, as dictated by the adaptive sampling method’s diurnal network pattern assumptions.

**On-demand Measurements**

We now describe on-demand measurement requests in NMIs. Recall from Chapter 1 that on-demand measurements are one-off requests that are used to quickly collect customized measurements for performance troubleshooting. For example, a network engineer might want to trace-back the sources of a DDoS attack as soon as possible by running on-demand measurement jobs over suspicious paths [46]. Hence, on-demand measurement requests need to be serviced by NMIs as early as possible or with least *response times*. We define response time as the time difference between the time when the measurement job is requested and the time when the request is finally served by the NMI. It is important to note that on-demand measurement jobs need to be served without affecting any on-going measurements. Using real-time systems terminology, an on-demand measurement request can be considered as an “online job” that needs to be scheduled in the time gaps between offline scheduled jobs. An online job is denoted by:

\[ J_k = (src_k, dst_k, tool_k, e_k). \]
2.3 Measurement Scheduling Challenges

In this section, we describe the various measurement scheduling challenges that need to be overcome in NMIs to handle the on-going and on-demand measurement requests.

2.3.1 Measurement Conflict Avoidance

The first challenge involves measurement conflict avoidance which can be explained as follows: Overlapping the execution intervals of two measurement jobs may or may not be problematic, depending on the measurement tools used. If a measurement tool is neither CPU intensive nor channel intensive (e.g. Ping or Traceroute), it does not interfere with other tools. Thus, overlapping its execution interval with others on the same server and/or path can still give us correct measurement reports. Such an overlap is called a “concurrent execution” with no conflict, which is desirable to improve the schedulability. On the other hand, active measurement tools such as Iperf, Pathload, Pathrate and Pathchar are CPU intensive since they use sophisticated calculations and/or channel intensive due to a large amount of probing packets as explained in Section 2.1. Thus, overlapping their execution intervals over the same measurement server or the same channel can cause serious interference and lead to misleading reports of the network status. We define execution overlap of multiple measurement jobs that results in misleading reports as a measurement conflict, which needs to be avoided while scheduling measurements.

Figure 2.4(a) shows an example of overlapping two measurement jobs $J_1$ and $J_2$: $J_1$ measures from $S_1$ to $S_2$ and $J_2$ measures from $S_2$ to $S_1$. If the tools of $J_1$ and $J_2$ do not conflict, i.e., they are neither CPU intensive nor channel intensive, the overlap is not a conflict. Such overlap is called a “concurrent execution” with no conflict, which is desirable
Figure 2.4: Scheduling active measurement tasks versus scheduling OS-threads and packets for improving the schedulability. However, if the overlap is a conflict, we must schedule $J_1$ and $J_2$ in a mutually exclusive way as shown in Figure 2.4(b). We note that our measurement-level scheduling problem is different from the traditional OS-level scheduling as depicted in Figure 2.4. Measurement-level scheduling determines the times when we start a measurement tool and when we stop it, not when the OS threads and packets are scheduled.
To illustrate the measurement conflict that produces misleading reports of network performance, we conduct an experiment. In the experiment, we connect two measurement servers by a LAN Testbed with 1500 Kbps bandwidth and run one H.323 videoconferencing session at 768 Kbps dialing speed as the background traffic. Thus, the remaining bandwidth should be approximately 732 Kbps. Given that streaming media and videoconferencing traffic is essentially UDP traffic, Iperf in UDP mode is popularly used to measure the available bandwidth. Thus, we make the two servers occasionally initiate Iperf jobs to monitor the available bandwidth. When we make two Iperf jobs run back-to-back with mutual exclusion (shown in the left-half of Figure 2.5), their measurements are in agreement with our expectation. However, when we intentionally make two Iperf execution durations overlap (shown in the right-half of Figure 2.5), it causes misrepresentation of the remaining bandwidth, merely due to conflicts of two Iperf jobs. This experiment thus shows that measurement tools that are CPU and/or channel intensive need to be initiated with proper orchestration to avoid measurement conflicts.

### 2.3.2 Measurement Regulation

In addition to the measurement conflict avoidance, measurement regulation is another challenge in measurement scheduling. Given the packet probing nature of measurement tools, a non-negligible amount of measurement traffic is generated into the network. If multiple measurements are initiated simultaneously, there is a possibility of these measurements utilizing excessive network resources. If these network resources are required by actual application traffic, the dearth of network resources due to measurement traffic consumption can significantly degrade the end-user QoE. Thus, measurement traffic needs to be regulated during measurement scheduling.
2.3.3 Centralized and Distributed Schedule Management

Figures 2.6(a) and 2.6(b) illustrate the centralized and distributed scheduling of measurement requests. In the case of centralized scheduling, a central regulator co-ordinates the measurement request gathering, overall schedule creation/modification, and also the final measurement data collection. In contrast, in distributed scheduling, measurement requests can arrive at any measurement server. Each server communicates with other servers to determine the overall schedule creation/modification and the final measurement data collection occurs at any of the measurement servers.

Centralized measurement scheduling is relatively more popular because it is convenient in NMIs to use a central regulator to initiate and collect measurements into a central database. Typically, the central regulator server also serves as the measurements analysis server that constantly updates a “network weather-map”, accessible via a web-interface. However, such a centralized schedule management can incapacitate an NMI when there is...
a failure of the central regulator. Further, some applications require distributed measurement scheduling without any intermediation by a central regulator. In [57], a measurement framework is proposed for a set of identical sensors distributed geographically. The sensors are given autonomy for data acquisition but they co-operate with each other to maintain global intelligence of the sensor network state. With such a distributed measurement framework, the analysis overhead is divided over the entire system instead of being imposed on a single sensor. In [58], several applications are described for distributed measurement using industrial instruments. The distributed measurement scheduling allows these applications to avoid a single point of failure and provides them with greater flexibility to dynamically determine the locations of measurement data collection and analysis. Hence, both centralized and distributed schedule management need to be addressed in NMIs.

### 2.3.4 Multi-domain Measurements

Collecting measurement data within a single ISP domain is not sufficient for troubleshooting performance bottlenecks that span network paths across multiple ISP domains. To understand the reason, let us consider Figure 2.7, which shows a multi-domain NMI for
provisioning end-to-end measurements across multiple paths spanning ISP domains: A, B, C, D and E.

Let us assume an end-user I connected to ISP A is experiencing intermittent video frames freezing when interacting via videoconferencing with end-user II connected to ISP B. In addition, let us assume the cause of the performance bottleneck between end-user I and end-user II to be a congested link in the intermediate path in ISP C’s domain. By suitably analyzing end-to-end measurements data collected between domains of ISPs A, C and E, we can identify the performance bottleneck link in ISP C’s intermediate network. For provisioning such inter-domain measurements, “NMI federations” [20] [21] are emerging. In these federations, multiple ISPs agree upon a common measurement policy to cooperate with each other for mutual benefit. The ISPs in the NMI federation implement common
techniques related to measurements initiation and measurement data exchange between their domains. Both the centralized and distributed schedule management methods can be used for multi-domain measurements. If using centralized scheduling, any particular domain maintains the central regulator and database. If using distributed scheduling, each domain maintains one or more databases for measurement data collection.

2.4 Round-robin Scheduling Algorithms

We now describe the existing scheduling algorithms that address some of the measurement scheduling challenges to handle on-going and on-demand measurement requests in NMIs. Since many of the earliest NMIs primarily collected Ping and Traceroute measurements which are neither CPU nor channel intensive, they did not have to pay attention to measurement conflict avoidance or measurement regulation. We refer to such a scheduling mechanism as “no-orchestration” measurement scheduling. However, many of today’s NMIs such as NLANR AMP [4], Internet2 E2EpiPES [21], NWS [14], Surveyor [59] and RIPE [6], employ toolkits that have several CPU and/or channel intensive measurement tools, which may cause measurement conflict problems.

To handle the measurement requests, [4] and [59] use a simple round-robin approach where measurement servers take turns such that only one tool executes at a time. In NMIs such as [21], a resource broker scheduling scheme is used. Using this resource broker scheme, multiple measurement requests are queued for scheduling and executed on a first-come first-serve basis on a measurement server. This scheme requires each tool in the toolkit of a measurement server to have its own resource broker. Due to above reasons, we can see that the resource broker scheme does not scale well when handling large number of measurement servers and tools. Further, it is not suitable for catering measurement
periodicity requirements. The network weather service (NWS) uses a token-passing mechanism [60] in an attempt to meet the pure periodicity measurement requirements while avoiding measurement conflicts. This mechanism allows only a single server in possession of a token to initiate measurements. The round-robin, resource broker, and token-passing are similar in principle, i.e., they allow only one instance of measurement to be executed at a time. As a result, they cannot leverage the concurrent execution of multiple measurement jobs, which limits schedulability and produce schedules with relatively long cycle times.

In addition, existing round-robin scheduling schemes do not have mechanisms to deliver least response times possible to on-demand measurement jobs that need to be serviced on top of on-going measurements. Another limitation of these schemes is that they do not use a systematic method for measurement requirements collection and schedule management. As a result, considerable time and manual effort needs to be expended to specify distinct sampling requirements, add or modify measurement tasks, and generate measurement schedules accordingly. Furthermore, the lack of a systematic method makes it hard to implement policy contracts for provisioning multi-domain measurements. However, both the resource broker and the token passing algorithms are inherently suitable for distributed measurement scheduling. In Chapter 3 of this dissertation, we present our measurement scheduling algorithms that more effectively address the measurement scheduling challenges compared to existing methods explained above.

2.5 Voice and Video over IP (VVoIP) Quality of Experience (QoE)

In this section, we describe the VVoIP system components and introduce the human, device and network factors that affect an end-user’s VVoIP. We also explain the streaming and interactive end-user VVoIP QoE types.
2.5.1 VVoIP System Overview

Figure 2.8 depicts the sender-side, network and receiver-side components in a VVoIP application (e.g. videoconferencing) session.

Sender-side Components

The combined voice and video traffic streams in the session are characterized by encoding rate \( b_{snd} \) originating from the sender-side. This device factor can be expressed as follows:

\[
b_{snd} = b_{voice} + b_{video} = tps_{voice} \left( \frac{b_{codec}}{ps} \right)_{voice} + tps_{video} \left( \frac{b_{codec}}{ps} \right)_{video}
\]  
\[
(2.1)
\]

where \( tps \) corresponds to the total packet size of either voice or video packets, whose value equals a sum of the payload size \( (ps) \) of voice or video packets, the IP/UDP/RTP header size (40 bytes) and the Ethernet header size (14 bytes); \( b_{codec} \) corresponds to the voice or video codec data rate values chosen. For high-quality videoconferences, G.711/G.722 voice codec and H.263 video codec are the commonly used codecs in end-points with peak
encoding rates of $[b_{\text{voice}}] = 64$ Kbps and $[b_{\text{video}}] = 768$ Kbps, respectively [61]. Although higher $[b_{\text{video}}]$ settings produce higher video quality, access link bandwidth limitations at the end-user site might not always permit the end-user to use high $[b_{\text{video}}]$ settings. Hence, end-points allow end-users to choose different $[b_{\text{video}}]$ settings depending on the desired video quality and the access link bandwidth at the end-user site. The end-users specify the $[b_{\text{video}}]$ setting as a “dialing speed” in a videoconference session. Dialing speeds of 768 Kbps and higher are chosen by end-users with access links of T-1 (1.5 Mbps upstream) or better, whereas, dialing speeds of 256 Kbps and 384 Kbps are chosen by end-users with cable-modem/DSL access links that have $\approx 500$ Kbps upstream bandwidth rates.

The $p$s of voice or video packets are influenced by the packetization at the sender-side before the packets are injected into the network. Owing to the sampling nature of voice codecs, voice streams contain a series of packets that are relatively small ($tps_{\text{voice}} \leq 534$ bytes) with fixed ps characteristics [62]. As for the video $p$s, they are mainly influenced by the activity-level ($a_{\text{lev}}$) in the video sequence at the sender-side. This is because most video codecs use inter-frame differencing encoding, where only frames containing differences between consecutive frames are sent rather than sending every video frame. The $a_{\text{lev}}$ refers to the temporal and spatial nature of the video sequences in a videoconference session. Consequently, higher the $a_{\text{lev}}$, higher are the frame rates or the packets per frame with each packet containing several video slices [63]. Broadly, video sequences can be categorized as having either low or high $a_{\text{lev}}$. Low $a_{\text{lev}}$ video sequences feature slow body movements and a constant background. High $a_{\text{lev}}$ video sequences feature rapid body movements and/or quick scene changes. Claire and Foreman [64] are examples of low and high $a_{\text{lev}}$ video sequences, respectively.
Network Components

Following the packetization process at the sender-side, the voice and video traffic streams traverse the intermediate hops on the network path to the receiver-side. While traversing, the streams are affected by the network factors i.e., end-to-end network bandwidth \( b_{net} \), delay \( d_{net} \), jitter \( j_{net} \) and loss \( l_{net} \) before they are collected at the receiver-side \( b_{rcv} \).

If there is adequate \( b_{net} \) provisioned in a network path to accommodate the \( b_{snd} \) traffic, \( b_{rcv} \) will be equal to \( b_{snd} \). Otherwise, \( b_{rcv} \) is limited to \( b_{net} \), whose value equals the available bandwidth at the bottleneck hop in the network path as follows:

\[
b_{rcv} = \min(b_{snd}, \min_{i=1..hops} b_{i\text{th hop}})
\]  

(2.2)

Receiver-side Components

The received audio and video streams are processed using dejitter buffers to smoothen the jitter effects and are further ameliorated using sophisticated decoder error-concealment schemes that recover lost packets using motion-compensation information obtained from the damaged frame and previously received frames. Finally, the decompressed frames are output to the display terminal for playback with an end-user QoE \( q_{mos} \) quantified in terms of a “Mean Opinion Score” (MOS) metric explained in Section 2.6.1.

2.5.2 Streaming and Interaction VVoIP QoE

End-users have different QoE expectations when using either one-way VVoIP applications such as video-on-demand or bidirectional VVoIP applications such as videoconferencing. To refer to the end-user satisfaction when passively viewing one-way VVoIP from a head-end, we use the term *streaming VVoIP QoE*. Similarly, to refer to the end-user satisfaction when interacting with another end-user using bidirectional VVoIP, we use
the term interaction VVoIP QoE. The end-user satisfaction is affected by voice and video impairments caused due to transient network fault events such as bandwidth bottlenecks and excessive packet delay, jitter or loss. The voice impairments include drop-outs, loudness and echoes. The video impairments include frame freezing, jerkiness, blurriness and tiling [31]. Figure 2.9 shows an example of video tiling on the site-B video in an actual videoconference. Such impairments present unwanted distractions to the end-users and thus degrade streaming VVoIP QoE.

The end-user satisfaction is also affected when participants in a videoconference experience interaction difficulties. The interaction difficulties correspond to instances during a session where a ‘listening’ participant is led to interrupt a ‘talking’ participant by saying “Can you please ‘repeat’ the previous sentence?” due to a voice drop-out caused by a network fault event occurrence. In extreme cases, prolonged network fault events impair the perceptual quality and aggravate the interaction difficulties to an extent that the participants
decide to ‘disconnect’ and ‘reconnect’ the videoconference session. The disconnect and re-
connect actions generally allow the participants to quickly recover from mis-configurations
of end-point jitter buffer and encoding rate settings during end-point adaptation to counter
the effects of such network fault events. Upon reconnection, assuming the network fault
events have subsided and end-point default settings have been restored, the participants ‘re-
orient’ their discussion to progress further with the remaining “agenda-items” to be covered
in the videoconference session. These ‘repeat’, ‘disconnect’, ‘reconnect’, and ‘reorient’
actions during a videoconference session are unwanted interaction patterns that obviously
degrade interaction VVoIP QoE.

2.6 Network QoS and VVoIP QoE Relation

In this section, we first describe the MOS metric that is used to quantify both streaming
and interactive VVoIP QoE. Next, we differentiate between the subjective and objective
QoE measurements. Lastly, we present existing objective VoIP and VVoIP QoE estimation
techniques and their short-comings.

2.6.1 Mean Opinion Score (MOS) Metric

The most commonly used metric to quantify end-user VVoIP QoE is the “Mean Opin-
ion Score” (MOS) standardized in ITU-T P.800. As shown in Figure 2.10, it has a scale
of 1 to 5 that maps to the different grades of end-user satisfaction. Specifically, the [4, 5]
MOS range corresponds to “Good” grade where an end-user perceives none or minimal
impairments and the application is always usable. The MOS [3, 4) range corresponds to
“Acceptable” grade where an end-user perceives intermittent impairments yet the applica-
tion is mostly usable. Lastly, the [1, 3) MOS range corresponds to “Poor” grade where
an end-user perceives severe and frequent impairments that make the application unusable.
Figure 2.10: MOS rankings for Good, Acceptable and Poor grades of user satisfaction

Thus, the aim of ISPs is to always sustain MOS rankings in Good grade levels \([4, 5]\) for optimal end-user VVoIP QoE.

Obtaining MOS rankings in a VVoIP system involves testing with several human subjects in a closed-network setting. It is relevant to note that the MOS rankings i.e., \(q_{mos}\), are obtained in the tests using a low or high \(a_{lev}\) video sequence for a set dialing speed \(b_{snd}\). The test cases comprise of a wide range of network conditions are configured using a WAN emulator such as NISTnet [65]. For these conditions, human subjects are asked to rank their QoE on the MOS scale. Hence, \(q_{mos}\) can be expressed as a function of the network factors as follows:

\[
q_{mos} = f(b_{net}, d_{net}, l_{net}, j_{net})
\]  

Earlier studies have shown that the \(q_{mos}\) can be expected to remain within a particular Good, Acceptable or Poor grade when each of the network factors are within certain performance levels shown in Table 2.6.1. Specifically, [66] and [67] suggest that for Good grade, \(b_{net}\) should be at least 20% more than the dialing speed value, which accommodates
additional bandwidth required for the voice payload and protocol overhead in a videoconference session; $b_{net}$ values less than 25% of the dialing speed result in Poor grade. The ITU-T G.114 [68] recommendation provides the levels for $d_{net}$ and studies including [3], [70] and [71] provide the performance levels for $f_{net}$ and $l_{net}$ on the basis of empirical experiments on the Internet.

<table>
<thead>
<tr>
<th>Network Factor</th>
<th>Good Grade</th>
<th>Acceptable Grade</th>
<th>Poor Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_{net}$</td>
<td>(&gt;922) Kbps</td>
<td>(576-922) Kbps</td>
<td>[0-576) Kbps</td>
</tr>
<tr>
<td>$d_{net}$</td>
<td>[0-150] ms</td>
<td>(150-300) ms</td>
<td>(&gt;300] ms</td>
</tr>
<tr>
<td>$l_{net}$</td>
<td>[0-0.5] %</td>
<td>(0.5-1.5) %</td>
<td>(&gt;1.5] %</td>
</tr>
<tr>
<td>$f_{net}$</td>
<td>[0-20] ms</td>
<td>(20-50) ms</td>
<td>(&gt;50] ms</td>
</tr>
</tbody>
</table>

Table 2.1: $q_{mos}$ GAP grades and performance levels of network factors for $[b_{video}] = 768$ Kbps

2.6.2 Subjective and Objective VVoIP QoE

The QoE rankings obtained from the closed-network testing explained in the previous section are referred to as subjective QoE rankings. Testing with human subjects is an expensive and time consuming process, and requires human subjects to be selected from diverse demographics and age groups. Hence, automated techniques that use psycho-acoustic/visual models of human perception are preferred to estimate objective QoE. The objective QoE rankings are also preferred for pro-active identification and troubleshooting of VVoIP performance bottlenecks since ISPs cannot rely on actual end-users for such purposes. However, the disadvantage in using objective QoE rankings is that they may not always accurately correlate with the actual QoE of end-users [72].
2.6.3 VoIP QoE Estimation Techniques

There are several objective techniques that use psycho-acoustic models to estimate VoIP QoE \cite{15} \cite{73} \cite{74} \cite{75}. The E-model \cite{15} is one such technique that has been repeatedly proven to be effective and thus has been widely adopted in measurement tools developed by industry (e.g., Telchemy’s VQMon \cite{76}) and open-source community (e.g., OARnet H.323 Beacon \cite{41}). Detailed studies of the E-Model can be found in works such as \cite{74} and \cite{77}. E-model’s success can be owed to its ability to provide online estimates of VoIP QoE based on instantaneous network health measurements (i.e., packet delay, jitter and loss) for a given voice encoding scheme. Earlier to E-Model, the only available techniques were offline techniques such as PESQ \cite{73} that are not suited for continuous monitoring of end-user VoIP QoE. The PESQ is an offline technique because it requires the source-side reference audio signal and its corresponding receiver-side audio signal that has experienced degradation due to network conditions.

The E-Model uses a psycho-acoustic R-factor scale whose values range from 0 to 100. The R-factor values can be mapped to MOS rankings as shown in Figure 10 2.11. The reasoning for such a mapping is described in detail in the ITU-T G.107 recommendation. In essence, the E-Model considers the effect of perceived voice impairments to be additive and expresses the VoIP QoE in terms of R-factor as follows:

\[
R = R_0 - I_s - I_d - I_e + A \approx 94 - I_d - I_e \quad (2.4)
\]

where, \( R_0 \) is the basic signal-to-noise ratio based on noise and loudness; \( I_s \) (Simultaneous Impairment Factor) represents the impairments occurring simultaneously with the speech signal, such as sidetone and PCM quantizing distortion; \( I_d \) (Delay Impairment Factor) represents the impairments that cause interaction difficulties and echoes; \( I_e \) (Equipment...
Impairment Factor) represents the impairments caused by transmission equipment that perform compression; $A$ (Advantage Factor) is used to capture the fact that end-users may be lenient in their rankings i.e., they will accept some quality degradation, in return for the ease of access, e.g. cell phone or satellite phone.

Although the E-Model is effective for online VoIP QoE measurements, the considerations used in the E-model are not pertinent for VVoIP QoE measurements due to idiosyncrasies in the video traffic encoding and impairment characteristics. The E-Model considers voice traffic as Constant Bit Rate (CBR) encoded with constant packet sizes and fixed data rates that are known for a given voice codec with a set audio sampling frequency. In comparison, the video traffic is Variable Bit Rate (VBR) encoded with variable packet sizes and bursty data rates that depend upon the temporal and spatial nature of the video sequences. Also, E-model considers voice signals to be affected by impairments such as
drop-outs, loudness and echoes whereas, video signals are affected by impairments such as frame freezing, jerky motion, blurriness and tiling [31].

2.6.4 VVoIP QoE Estimation Techniques

To estimate VVoIP QoE, there are several previous objective techniques such as [78] and [79]. These techniques estimate VVoIP QoE by performing frame-to-frame Peak Signal-to-Noise Ratio (PSNR)\(^2\) comparisons of the original video sequence and the reconstructed video sequence obtained from the sender-side and receiver-side, respectively. PSNR for a set of video signal frames is given as follows:

\[
PSNR(n)_{db} = 20\log_{10}\left(\frac{V_{\text{peak}}}{\text{RMSE}}\right)
\]  

(2.5)

where, signal \(V_{\text{peak}} = 2^k - 1\); \(k\) = number of bits per pixel (luminance component); \(\text{RMSE}\) is the mean square error of the \(N^{th}\) column and \(N^{th}\) row of sent and received video signal frame \(n\). Thus obtained PSNR values are mapped to MOS rankings as shown in Figure 2.12.

The PSNR-mapped-to-MOS technique is an offline technique because it is computationally intensive in its per-pixel processing and also requires time and spatial alignment of the original and reconstructed video sequences. Hence, it is not useful for measuring online VVoIP QoE on operational network paths. It is typically used for purposes such as pre-deployment assessments of sustainable VVoIP traffic loads on networks and codec performance testing in labs. In addition, the PSNR-mapped-to-MOS technique does not address impact of the joint degradation of voice and video frames. Hence, impairments that annoy end-users such as “lack of lip synchronization” [80] due to voice trailing or leading

\(^2\)PSNR is a derivative of SNR; it compares the maximum possible signal energy to the noise energy. PSNR has shown higher correlation with perceptual quality assessments from actual end-users than the conventional SNR [32].
Figure 2.12: Mapping of PSNR values to MOS rankings

video are not considered in the VVoIP QoE estimation. Further, the traditional PSNR-mapped-to-MOS technique has been proven to be inaccurate in terms of correlation with perceived visual quality in many cases [81] [82] [83]. The cause for inaccuracy is the non-linear behavior of the human visual system for compression impairments. To overcome these limitations, several modifications have been made to the traditional PSNR-mapped-to-MOS technique to improve estimation accuracy. The improved PSNR-mapped-to-MOS technique has been ratified by communities such as the ITU-T in their J.144 Recommendation [79] and the ANSI in their T1.801.03 Standard [84].

To obtain online VVoIP QoE estimates, there have been recent attempts to develop suitable techniques in works such as [63], [85] and [86]. In [63], video distortion due to packet loss is estimated using a loss-distortion model. The loss-distortion model uses online loss measurements and takes into account other inputs such as video codec type, coded bit rate and packetization to estimate online PSNR values. The limitation of this work
is that the PSNR degradation is not compared with subjective assessments from actual human subjects and hence the approach effectiveness is questionable. In [85], a Human Visual System (HVS) model is proposed that produces video QoE estimates without requiring reconstructed video sequences. This study validates their estimation accuracy with subjective assessments from actual human subjects, however, the HVS model is primarily targeted for 2.5/3G networks. Consequently, it only accounts for PSNR degradation for online measurements of noisy wireless channels with low video encoding bit rates. In [86], a random neural network (RNN) model is proposed that takes video codec type, coded bit rate, packet loss as well as loss burst size as inputs and produces online MOS estimates. All of the above models do not address impact of human interaction that is affected by excessive network delay and jitter. Further, these studies do not address the joint degradation of voice and video frames in the end-user VVoIP QoE estimation. In comparison, our proposed techniques presented in Chapter 4 address both these issues, as well as the other salient issues in online VVoIP QoE estimation.

2.7 Summary

In this Chapter, we defined several terms and presented concepts that are repeatedly used in the remainder of this dissertation. More specifically, we defined the various network QoS metrics and the active probing techniques that have been implemented in measurement tools of NMIs to measure network QoS. We also defined on-going and on-demand measurements that ISPs use for network control and management functions which aim to improve end-user QoE. We illustrated the measurement conflict problem and motivated the need for measurement regulation, centralized and distributed measurement schedule management, and provisioning multi-domain measurements. We then presented an overview
and limitations of existing network measurement scheduling techniques. Following this, we shifted our attention on analysis techniques that estimate end-user QoE of VVoIP applications based on the provisioned network QoS measurements. We discussed about the two end-user VVoIP QoE types i.e., streaming and interactive in a VVoIP system. Finally, we explained how existing techniques such as the ITU-T E-Model and ITU-T J.144 estimate end-user VoIP and VVoIP QoE in terms of MOS rankings, respectively.
In this chapter, we present our measurement scheduling algorithms that can handle ongoing and on-demand measurements that span multiple ISP domains. The algorithms use a Concurrent Execution (CE) principle explained in Section 3.1 that improves schedulability without violating the measurement conflict constraint. In Section 3.2, we explain the Measurement Level Agreement (MLA) that specifies the amount of measurement regulation on a network path by the measurement scheduling algorithms. Next, in Section 3.3, we present the pseudo code of our measurement scheduling algorithms. Our first algorithm is a heuristic bin packing scheduling algorithm that uses the CE principle to improve the schedulability in offline measurement scheduling. Although this algorithm is suitable for routine network monitoring purposes, it does not cater to the measurement periodicity required for accurate network weather forecasting. For high schedulability as well as for maintaining periodicity, we present our second algorithm that is a variant of the Earliest Deadline First (EDF) scheduling algorithm. We refer to this scheduling algorithm as EDF-CE. Following this, we present our third algorithm referred to as the recursive-push algorithm. This algorithm schedules on-demand measurement requests with minimum response time on top of the offline schedules generated by the EDF-CE algorithm.
While explaining the three algorithms in Section 3.3, for simplicity reasons, we assume centralized scheduling i.e., we assume a central regulator collects the on-going and on-demand measurement requests and handles them using our scheduling algorithms. We also assume that our scheduling algorithms will be used within a single ISP’s domain. In Section 3.4, we present a distributed scheduling mechanism that does not require a central regulator and hence avoids incapacitating an NMI upon occurrence of a central regulator failure. In Section 3.5, we extend the scheduling algorithms to function across multiple ISP-domains. In particular, we describe technical and policy issues in active measurement federations and explain how they allow inter-domain measurement scheduling for provisioning multi-domain measurements. In the last part of this chapter we present two case studies that assess the utility of our measurement scheduling algorithms. The first case study presented in Section 3.6, assesses the utility in the ActiveMon framework, which is used for routine network monitoring. The second case study presented in Section 3.7, assesses the utility in the Network Weather Service (NWS) framework, which is used for network weather forecasting.

3.1 Concurrent Execution (CE) Principle

As explained in Section 2.4 of Chapter 2, existing scheduling algorithms run only one measurement at a time in the NMI. This severely limits the schedulability of measurement requests and affects frequent sampling of network status. To overcome this limitation, we propose a “concurrent execution” (CE) principle which allows concurrent execution or overlapping of non-conflicting measurements in the measurement schedules if:

(a) measurements involve tools that are neither CPU and/or channel intensive, and

(b) measurements involve network paths that do not have common links.
Such an execution of measurements also avoids measurement conflicts, while improving schedulability.

To implement the CE principle in measurement scheduling, we construct a task conflict graph by combining the measurement topology $G$ and the task set $\Gamma$. Figure 3.1 shows an example task set. For the given measurement topology and the task set in Figures 3.1(a) and (b), we examine each pair of tasks $\tau_i$ and $\tau_j$ to see if they share the same source server, destination server or part of the paths between source and destination servers. If so, the two tasks may “potentially” conflict if scheduled concurrently. In Figures 3.1(a) and (b), $\tau_1$ and $\tau_2$ share $S_2$ and thus we add a dependency edge between them in the potential task conflict graph as in Figure 3.1(c). $\tau_2$ and $\tau_3$ share the path and thus another dependency edge is added. On the other hand, $\tau_1$ does not share any network resource with $\tau_3$ and so no edge is added in this case. Even if two tasks share network resources, they may not
actually conflict depending on the active measurement tools used. Based on our empirical studies in [87], we could determine which tools conflict if they run concurrently. The result is summarized by the tool conflict matrix in Figure 3.1(d). For example, Iperf and Pathchar conflict if they run concurrently on the same server since both intensively use server and channel resources for active measurement. On the other hand, Ping just injects small probing packets and hence does not conflict with any other tools. Considering the tool conflict matrix, the potential task conflict graph in Figure 3.1(c) can be converted to the final task conflict graph of Figure 3.1(e). The edge between two tasks in the final task conflict graph indicates that they should be scheduled in a mutually exclusive manner to avoid measurement conflicts.

3.2 Measurement Level Agreement (MLA)

For regulating the amount of measurements that need to be scheduled, we propose using a “Measurement Level Agreement” (MLA) specification in the NMI. The MLA specifies the percentage (1% - 5%) or only bits per second (1 Mbps - 2 Mbps) of the network bandwidth in ISP backbones that could be used for measurement traffic\(^3\). We use the notation \(\psi\) to denote the MLA specification.

In the measurement-level scheduling problem, the sum of the bandwidth usage by concurrent measurement jobs over the same channel should be less than \(\psi\) at all times. To better understand this concept, we refer to Figure 3.2, which shows an example measurement schedule whose task conflict graph is shown in Figure 3.1(e). The schedule assumes \(\tau_{11}\) and \(\tau_{12}\) are measurement jobs correspond to task \(\tau_1\), \(\tau_{21}\) job corresponds to task \(\tau_2\), \(\tau_{31}\) since most active measurement tools have options to specify packet sizes and bandwidth usage of a measurement test, simple calculations can be used to determine how much of a network’s bandwidth will be used by a given set of active measurements, over a certain period of time.
Figure 3.2: Example measurement schedule satisfying MLA $\psi$

and $\tau_{32}$ are jobs correspond to task $\tau_3$, and $\tau_{41}$ job corresponds to task $\tau_4$. It also assumes each job consumes 1 Mbps of network bandwidth and the specified MLA regulation constraint $\psi = 2$ Mbps. Hence, concurrent execution of $\tau_{11}$ and $\tau_{31}$ jobs of non-conflicting tasks (no edge between $\tau_1$ and $\tau_3$) will satisfy the MLA. However, scheduling $\tau_{41}$ job concurrently with $\tau_{11}$ and $\tau_{31}$ violates the MLA, even though allowed as per CE principle (no edge between $\tau_1$, $\tau_3$ and $\tau_4$). Hence, $\tau_{41}$ is placed in the next time slot of the measurement schedule. Using a similar reasoning, jobs $\tau_{12}$, $\tau_{21}$ and $\tau_{32}$ are scheduled in Figure 3.2 using the CE principle and without violating the MLA specification.

### 3.3 Scheduling Algorithms

In this section, we first explain our heuristic bin packing scheduling algorithm that improves measurement schedulability but does not cater to any periodicity requirements during offline schedule creation. Next, we explain our EDF-CE scheduling algorithm that improves measurement schedulability and also addresses periodicity requirements during offline schedule creation. Lastly, we explain our recursive-push scheduling algorithm that schedules on-demand measurements with minimum response time, without disrupting the offline EDF-CE scheduled measurements.
3.3.1 Heuristic Bin Packing Scheduling Algorithm

In order to serve the problem of minimizing the cycle time, we develop a “heuristic bin packing” algorithm based on the classic bin packing algorithm used in real-time systems. The flowchart of the algorithm is shown in Figure 3.3. The algorithm packs the measurement jobs corresponding to a measurement task $\tau_i$ in schedule bins using a “first-fit decreasing” [95] principle. The first-fit decreasing suggests that jobs are ordered with decreasing execution times, and jobs with higher execution times are scheduled earlier than jobs with relatively lower execution times. A schedule bin corresponds to a time fraction of the cycle time within which measurement jobs are packed i.e., scheduled. While scheduling the jobs in this way, the heuristic bin packing algorithm uses the CE principle and adheres to the specified MLA. The output is the measurement schedule i.e., start times and finish times of all the scheduled measurement jobs.
Performance Evaluation

Herein, we show the improvement in cycle time obtained by using the heuristic bin packing algorithm compared to the round-robin scheduling algorithm used in NMIs such as [4] [59] [21] [60]. For this, we use an example problem in which we consider a task set comprising of tools that have small (5 minutes) and relatively longer (20 minutes) execution times. The offline measurement tasks are specified for a set of measurement servers $N = \{A,B,C,D,E,F\}$ using different measurement topologies: full-mesh, tree and hybrid topologies. Figures 3.4 (a) - (c) show the full-mesh, tree and hybrid measurement topologies, respectively. A full-mesh topology involves testing from each measurement server to every other measurement server. A tree topology involves testing between neighboring measurement servers in the topology. Lastly, a hybrid topology involves supporting subsets of measurement servers that need full-mesh and tree topology measurements.

Figure 3.5 shows the simulation results for cycle times obtained, with the increase in the number of measurement servers. For this simulation, we choose a fixed bin size of 20 minutes. We can observe the notable savings in cycle time are obtained for the full-mesh topology by using the heuristic bin packing instead of the round robin packing for 5 measurement servers and beyond. The savings with the increase in number of measurement
Figure 3.5: Cycle time comparison for round-robin and heuristic bin packing

Figure 3.6: Effects of increase in number of measurement tools on cycle times in heuristic bin packing
servers are such that cycle times increase linearly in the case of heuristic bin packing, whereas they increase exponentially in the case of round robin packing. This is because, the round robin packing does not leverage the CE principle, which the heuristic bin packing uses. The savings in cycle time for the tree topology, though are noticeable, they are not as significant as in the case of the full-mesh topology. This is due to the fewer jobs that are handled in the tree topology case. Our simulations for hybrid topology showed results in accordance with the results obtained for the full-mesh and tree topology. For a hybrid topology with a dominant number of full-mesh topology connected measurement servers, the savings in cycle time are significant. However, for a hybrid topology with a dominant number of tree topology connected measurement servers, the savings in cycle time are relatively less. In any case, the heuristic bin packing outperforms the round robin packing in terms of cycle time savings for any given measurement topology.

Figure 3.6 shows the simulation results obtained for a full-mesh topology, with a fixed bin size of 20 minutes. As the number of measurement tools increase for a given number of measurement servers (5, 10 or 15), the cycle time increase has a notable trend. The cycle time increase is almost exponential in the case of round robin packing whereas the cycle time increase is almost linear in the case of the heuristic bin packing. Thus, 100% till up to 300% savings in cycle time can be obtained using the heuristic bin packing when 5 or more measurement servers are used in the NMI. From these results, we can in summary infer that the heuristic bin packing significantly reduces the cycle times and outperforms round robin packing for 5 or more measurement servers and any number of measurement tools used in the NMI.

Figure 3.7 shows the simulation results obtained for a full-mesh topology with varying bin sizes. The results show that choosing different bin sizes can affect the heuristic bin
Figure 3.7: Effects of bin sizes on cycle times in heuristic bin packing

packing patterns, and thus the cycle times. In our example that is used in the simulation, choosing a bin size of 20, 40 or 60 results in a much less cycle time than choosing a bin size of 35. This is true irrespective of the number of measurement servers for which measurement jobs need to scheduled. Obviously, if the execution times of the jobs are chosen differently, a similar simulation methodology can be adopted to determine the optimum bin sizes that produce the most savings in cycle time. Hence, choosing an optimum bin size for a given set of tasks can further improve the cycle time savings obtainable using heuristic bin packing compared to the round robin packing.

3.3.2 Offline EDF-CE Scheduling Algorithm

To derive the offline schedule obtained through EDF-CE computations that cater to the periodicity requirements, we use the example task set and final task conflict graph shown in Figure 3.1. One obvious solution is to start a measurement job at the source server at its
release time without considering measurement conflict and MLA constraints. Figure 3.8 shows such a schedule. The upward arrows in the figure indicate the release times of the periodic measurement tasks. The schedule, however, causes a number of conflicts that result in misleading report of the actual network performance. Another approach is to run only a single measurement job at any time instant using a non-preemptive EDF scheduling algorithm. Figure 3.9 shows such a schedule for the same problem. It can completely prevent conflicts. However, it does not allow concurrent execution of multiple jobs even if they do not conflict, which degrades the schedulability.

Our aim to find a schedule in between these two extremes such that conflicts are completely prevented while maximizing the concurrent execution whenever possible. For this, we propose the EDF-CE (i.e., EDF with Concurrent Execution) algorithm that schedules measurement jobs in the EDF order while allowing concurrent execution if jobs do not conflict. The algorithm is formally described in the following:

The EDF-CE algorithm maintains the ordered list of release times $rt\_list$ and the ordered list of finish times $ft\_list$. Line 1 initializes $rt\_list$ with all release times in a hyper-period. In Figure 3.10, the release times are 0, 20, 30, 40, 60, 80, 90, 100, and 120. Line
Algorithm 1 EDF-CE: given a task conflict graph, find measurement schedule in a hyperperiod

1: \textbf{Input:} task set $\Gamma$ and task conflict graph
2: \textbf{Output:} start time $st_{ij}$ and finish time $ft_{ij}$ for each job $\tau_{ij}$ in a hyperperiod
3: \textbf{begin procedure}
4: Initialize $rt\_list$ with the ordered list of all release times in a hyperperiod
5: Initialize $ft\_list = \{\}$ /* ordered list of finish times*/
6: Initialize $pending\_job\_queue = \{\}$
7: \textbf{repeat}
8: \hspace{1em} $time = \text{get the next scheduling time point from } rt\_list \text{ and } ft\_list$
9: \hspace{1em} add all newly released jobs at $time$ to $pending\_job\_queue$ in EDF order
10: \hspace{1em} \textbf{for} each job $\tau_{ij}$ in $pending\_job\_queue$ in EDF order \textbf{do}
11: \hspace{2em} if $\tau_{ij}$ does not conflict with any of already scheduled jobs at $time$ and scheduling $\tau_{ij}$ at $time$ does not violate MLA constraint $\psi$
12: \hspace{3em} $st_{ij} = time$ and $ft_{ij} = time + e_i$
13: \hspace{2em} if $ft_{ij}$ is later than the deadline of $\tau_{ij}$
14: \hspace{3em} return error /* infeasible task set */
15: \hspace{2em} \textbf{end if}
16: \hspace{2em} remove $\tau_{ij}$ from $pending\_job\_queue$
17: \hspace{2em} add $ft_{ij}$ to $ft\_list$ in order
18: \hspace{2em} \textbf{else}
19: \hspace{3em} do nothing /* $\tau_{ij}$ will be considered again at the next scheduling time point in the outer loop */
20: \hspace{2em} \textbf{end if}
21: \hspace{1em} \textbf{end for}
22: \hspace{1em} \textbf{until} $time == \text{hyperperiod}$
23: \textbf{end procedure}
Figure 3.10: EDF-CE Schedule
2 initializes $ft_{list}$ as empty since no job is scheduled yet. Note that the only time points when we need to make a scheduling decision are either when a new job is released or a current executing job is finished. Thus, we call times in $rt_{list}$ and $ft_{list}$ “scheduling time points”. In addition, the algorithm maintains a $pending_{job\_queue}$ that holds all jobs released but not scheduled, in the EDF order. Line 3 initializes it as empty. The do-until loop from Line 4 to Line 20 progresses the virtual time variable $time$ upto a hyperperiod while determining the schedule at all scheduling time points. Line 5 moves $time$ to the next scheduling time point. Then, Line 6 adds all newly released jobs to the $pending_{job\_queue}$.

The for loop from Line 7 to Line 19 examines the pending jobs in the EDF order and determines whether they can start at $time$ without causing any conflict and without violating MLA $\psi$ (see Lines 8 and 9). If so, the job $\tau_{ij}$’s start time $st_{ij}$ is determined as $time$ and its finish time $ft_{ij}$ is determined as $time + e_i$ in Line 10. If the finish time $ft_{ij}$ is later than the deadline of job $\tau_{ij}$ in Line 11, we cannot construct a feasible schedule meeting all deadlines and hence return error in Line 12. If we can meet the deadline of $\tau_{ij}$, we can continue. In Line 14, $\tau_{ij}$ is removed from the $pending_{job\_queue}$. Also, its finish time $ft_{ij}$ is added to $ft_{list}$ so that $ft_{ij}$ can be considered as a new scheduling time point in the outer do-until loop. If $\tau_{ij}$ cannot be scheduled at $time$ (Line 16), it is kept in the $pending_{job\_queue}$ and can be considered again at the next scheduling time point by the outer loop. Note that the algorithm tries to concurrently start as many jobs as possible in the EDF order at $time$ as long as they neither conflict nor violate the MLA. Figure 3.10(a) shows such EDF-CE schedule for the same problem of Figure 3.1. At time 0 of the EDF-CE schedule, note that $\tau_{11}$, $\tau_{31}$, and $\tau_{41}$ are executed concurrently but not $\tau_{41}$, which is maximizing the concurrent execution guaranteeing no-conflict.
Once we find the EDF-CE schedule, we can convert it to the measurement schedule table of each server considering the source server of each job. Figure 3.10(b) shows the schedule tables of all the five servers. Such constructed schedule tables are transferred to corresponding servers so that they can start and stop the measurement jobs in a timely manner.

**Performance Evaluation**

We evaluate the performance of our EDF-CE scheduling algorithm using synthetic measurement tasks to show the maximum schedulability. Our synthetic task set is comprised of four periodic active measurement tasks $\tau_1$, $\tau_2$, $\tau_3$ and $\tau_4$. The period $p_i$ of each task $\tau_i$ is randomly generated from [1000 sec, 10000 sec]. The execution time $e_i$ of each task $\tau_i$ is randomly generated from [100 sec, 999 sec]. Since the measurement topology and inter-task conflict relations can be represented by a task conflict graph, we conduct this experiment by changing only the task conflict graph. The task conflict graph of the four tasks is randomly created using a parameter called a conflict factor. The conflict factor represents the probability that there is a conflict edge between any two tasks. Therefore, when the conflict factor is 1, the task conflict graph is fully connected. If the conflict factor is 0, there is no edge between tasks.

For each sample task set and task conflict graph, we use the “maximum schedulable utilization” $\sum_{i=1}^{4} e_i/p_i$ as the performance metric. We determine the maximum schedulable utilization by gradually increasing execution times $e_i$ until the scheduling algorithms fail to construct a feasible schedule.
We compare three scheduling algorithms:

(a) No-orchestration used in traditional NMIs that scheduled measurement jobs at their release times without considering measurement conflicts,

(b) EDF that schedules only one measurement job at a time using the non-preemptive EDF algorithm just like a single-processor EDF scheduling that is equivalent to the round-robin scheduling used in NMIs such as [4] [59] [21] [60], and

(c) our proposed EDF-CE.

Figure 3.11 shows the maximum schedulable utilization with the increase of the conflict factor. Here, we assume a large MLA $\psi$, say 25 Mbps, and thus avoid any MLA bottlenecks when finding the schedule. Each plotted point in the figure is the average of 1000 random sample task sets. EDF’s maximum schedulable utilization is constantly bounded
under 100% regardless of the conflict factor since it does not allow concurrent execution even when possible. On the other hand, our EDF-CE algorithm can maximally utilize the concurrent execution whenever possible. When the conflict factor is zero, EDF-CE allows concurrent execution of all four tasks. This is similar to scheduling the four tasks on four independent processors. Thus, the maximum schedulable utilization reaches up to 400%. As the conflict factor increases, the maximum schedulable utilization gradually decreases. When the conflict factor is 1, i.e., when all four tasks conflict with each other, EDF-CE automatically degenerates to the single processor EDF and hence gives the maximum schedulable utilization of 100%. The result shows that: (a) EDF-CE is leveraging the “maximal but only possible” concurrent execution by explicitly considering the conflict dependency among tasks, (b) at low conflict factor conditions, upto 100% improvement in maximum schedulable utilization can be obtained for every pair of non-conflicting tasks. The no-orchestration approach always gives the maximum schedulable utilization of 400%
since all four tasks can be concurrently executed ignoring the conflict dependency. This, however, causes many conflicts as will be shown in Section 3.6.2, resulting in many misleading reports of actual network performance.

Figure 3.12 illustrates how the maximum schedulable utilization of EDF-CE is bounded by the MLA constraint $\psi$ and conflict factor. As expected, a higher value of $\psi$ accommodates a larger number of concurrent jobs and hence produces a higher maximum schedulable utilization. For a given $\psi$ value, the maximum schedulable utilization is constant up to a certain point of the conflict factor and then starts decreasing. Such a trend explains that $\psi$ is the bottleneck when the conflict factor is small, whereas the conflict dependency becomes the bottleneck when the conflict factor is large. Further, when the conflict factor is large and the $\psi$ is set to a low value that further restricts any possible concurrent execution, EDF-CE may only produce less than 100% improvement in the maximum schedulable utilization when compared to the EDF.

3.3.3 Online Recursive-push Scheduling Algorithm

At the run time, while each server executes periodic measurement tasks according to the pre-computed schedule table, a network engineer can request an on-demand measurement $J_k$. For now, we assume that such a request is received by the central regulator.

Upon the arrival of an on-demand request $J_k = (src_k, dst_k, tool_k, e_k)$, our goal is to serve it as early as possible without missing any deadlines of periodic measurement tasks. For this, we propose a recursive push algorithm that recursively pushes offline scheduled periodic jobs within their deadlines. This push can create a left-over time called a slack as early as possible and this slack time can be used to schedule $J_k$. The basic idea of recursive push can be best illustrated by Figure 3.13 that shows the same EDF-CE schedule as above.
Suppose that an on-demand request $J_k = (S_2, S_3, Iperf, 10)$ arrives at time 50. We assume that $J_k$ conflicts with $\tau_1$, $\tau_2$, and $\tau_3$ as shown by the modified task graph. The central regulator cannot allow $J_k$ to start at its arrival time 50 since it conflicts with $\tau_{22}$. Thus, the central regulator calculates the maximum slack from starting at 55. For this, the central regulator calls \texttt{push($\tau_{12}$)} and \texttt{push($\tau_{34}$)} to determine how much $\tau_{12}$ and $\tau_{34}$ can be pushed to make the maximum slack for $J_k$. The \texttt{push} operation is recursive. To determine the maximum \texttt{push} of $\tau_{12}$, we first have to know the maximum \texttt{push} of the dependent job $\tau_{23}$. Thus, \texttt{push($\tau_{12}$)} recursively calls \texttt{push($\tau_{23}$)} and in turn \texttt{push($\tau_{23}$)} calls \texttt{push($\tau_{36}$)}. On the other hand, $\tau_{36}$ does not conflict with any other offline scheduled jobs while being pushed up to its deadline $d_{36} = 120$. Such a job with which the recursion can terminate is called a \textit{terminal job}. Similarly, $\tau_{34}$ is also a terminal job. For a terminal job $\tau_{ij}$, the
The **push** procedure can determine its new pushed finish time $new_{ft_{ij}}$ and new pushed start time $new_{st_{ij}} = new_{ft_{ij}} - e_i$ without any further recursive calls. The **push** operation is formally defined as follows:

**Algorithm 2 push:** return the new start time of input jobs after maximum push

1: **Input:** $\tau_{ij}$
2: **Output:** new start time after maximum push $new_{st_{ij}}$
3: **begin procedure**
4: if $\tau_{ij}$ has no conflicting jobs scheduled up to $d_{ij}$ then
5: /* terminal job */
6: slide $\tau_{ij}$ from $st_{ij}$ to $d_{ij} - e_i$ until MLA violation is observed at $t_{MLA}$ ($t_{MLA} < d_{ij}$).
7: if $t_{MLA}$ is found, the new finish time $new_{ft_{ij}} = t_{MLA}$. otherwise, $new_{ft_{ij}} = d_{ij}$.
8: new start time $new_{st_{ij}} = new_{ft_{ij}} - e_i$.
9: else
10: /* not a terminal job */
11: new finish time $new_{ft_{ij}} = d_{ij}$.
12: for each conflicting task $\tau_{ij'}$ up to $d_{ij}$
13: $new_{ft_{ij}} = \min (new_{ft_{ij}}, push(\tau_{ij'}))$.
14: end for
15: slide $\tau_{ij}$ from $st_{ij}$ to $new_{ft_{ij}} - e_i$ until MLA violation is observed at $t_{MLA}$ ($t_{MLA} < new_{ft_{ij}}$).
16: if $t_{MLA}$ is found, the new finish time $new_{ft_{ij}} = t_{MLA}$. otherwise keep $new_{ft_{ij}}$.
17: new start time $new_{st_{ij}} = new_{ft_{ij}} - e_i$.
18: end if
19: return $new_{st_{ij}}$.
20: **end procedure**

This algorithm returns the new start time $new_{st_{ij}}$ after maximally pushing $\tau_{ij}$. If $\tau_{ij}$ is a terminal job, its new finished time can be pushed up to its deadline $d_{ij}$ if we could ignore the MLA constraint. In order to consider the MLA constraint, in Line 2, we slide $\tau_{ij}$’s execution interval up to $d_{ij}$ to find a earliest time point $t_{MLA}$ when the MLA constraint can be violated, if any. If such time point $t_{MLA}$ is found, $t_{MLA}$ is the latest possible pushed finish time of $\tau_{ij}$ without violating the MLA constraint. Thus, $new_{ft_{ij}}$ is set to $t_{MLA}$ in Line 3. Otherwise, the new finish time can be pushed up to $d_{ij}$, that is, $new_{ft_{ij}} = d_{ij}$ in
Line 3. Once the new finish time is determined, Line 4 can simply calculate the new start time, i.e., \( \text{new\_st}_{ij} = \text{new\_ft}_{ij} - e_i \).

If \( \tau_{ij} \) is not a terminal job, Lines 7, 8, and 9 recursively call \text{push} for all dependent jobs to figure out the minimum new start time of all dependent jobs. If we ignore the MLA constraint, the minimum of the deadline \( d_{ij} \) and the new pushed start times of all dependent jobs is the latest possible new finish time \( \text{new\_ft}_{ij} \) for \( \tau_{ij} \). Lines 10 and 11 can advance the new finish time \( \text{new\_ft}_{ij} \) considering the MLA constraint in the same way as in the terminal job case. With \( \text{new\_ft}_{ij} \), Line 12 calculates the new start time as \( \text{new\_st}_{ij} = \text{new\_ft}_{ij} - e_i \). Finally, Line 14 returns \( \text{new\_st}_{ij} \).

Considering \( \text{new\_st}_{ij} \) of all dependent jobs of \( J_k \), we can calculate the maximum slack that can be used for the on-demand request \( J_k \) starting from the current scheduling time point \( t \). If the maximum slack is larger than the required execution time \( e_k \) and also if executing \( J_k \) from \( t \) to \( t + e_k \) does not violate the MLA constraint, the central regulator sets time \( t \) as the start time of \( J_k \) and push dependent periodic jobs as needed. The piece of schedule affected by \( J_k \) (see “schedule replacement” in Figure 3.13) is transferred to the corresponding servers so that they can temporarily use the updated schedule piece instead of the original schedule, to accommodate \( J_k \). If the above condition does not hold, the central regulator examines the next scheduling time point to recalculate the maximum slack and so on, until it finds enough slack time during which \( J_k \) can be executed without violating the MLA constraint.

**Performance Evaluation**

We evaluate the performance of our recursive push scheduling algorithm using synthetic measurement tasks. We show the improvement in average response times of on-demand requests by simulating on-demand jobs with random arrivals and scheduling them over the
offline EDF-CE schedule. The offline specified task set consists of four periodic tasks as before in the EDF-CE case, and their execution times and periods are randomly generated from [1 minute, 10 minutes] and [20 minutes, 200 minutes], respectively. The execution times and inter arrival times of on-demand jobs are also randomly generated from [1 minute, 10 minutes] and [20 minutes, 200 minutes], respectively. The performance metric is the average of the response times for 1000 on-demand jobs. We compare our recursive push algorithm with the background approach, which is a naive and commonly used approach in NMIs. This approach schedules an on-demand job in the earliest gap present in the offline EDF-CE schedule within which the on-demand job can execute to completion.

Figure 3.14 shows that our recursive push algorithm can significantly improve the responsiveness for on-demand measurement requests. Note that the difference in the average response times of the background and recursive push cases increases as the conflict factor increases. This is because a higher conflict dependency among tasks reduces the concurrent execution of jobs and thus reduces the gaps available to schedule the on-demand jobs.

To estimate the overhead of online scheduling, we measure the algorithm running time for each on-demand job on 2.4 GHz Pentium 4 Linux PC. Figure 3.15 shows the average times as increasing the number of periodic tasks, while fixing the conflict factor as 0.8. Even for a large number of periodic tasks with a high conflict factor, our recursive push algorithm can find the slack and calculate the updated schedule within tens of milliseconds. This is a negligible delay comparing with typical measurement task execution times in the order of minutes.
Figure 3.14: Average response time of on-demand jobs

Figure 3.15: Online schedule overhead for on-demand jobs
3.4 Distributed Implementation of Scheduling Algorithms

The offline and online scheduling algorithms described in the previous sections assume a central regulator that collects all offline/online measurement requests and builds/updates the global schedule. In a distributed setting, measurement requests (e.g., add/remove periodic measurement tasks and on-demand measurement jobs) arrive at any of the NMI’s measurement servers, possibly concurrently. If each server concurrently updates the schedule upon the arrival of requests, it breaks the consistency of the schedule and in turn creates measurement conflicts. Therefore, the issue is to serialize the distributed concurrent requests such that the schedule can be updated in a consistent way. For this, we propose to use Raymond’s algorithm [29] developed for distributed synchronization. This section describes how Raymond’s algorithm works with our scheduling algorithms maintaining the schedule consistency in a distributed way.

For the measurement topology given in Figure 3.16 (a) as an example, we first create the minimal spanning tree as in Figure 3.16 (b). This tree is used to maintain a tree-wide single lock with minimal exchange of messages [29]. The basic idea is to allow only the lock holder to commit the arrival of a request at a time, which assures the global serialization.

Figure 3.16: Minimal spanning tree for the measurement server topology
of concurrent requests. In the initialization phase, we place the lock at any server, say $S_2$ in the example of Figure 3.16 (b), and make each server set its $dir$ variable to the neighbor toward the lock holder as shown in Figure 3.16 (c).

Upon the arrival of a new request at a server, the server exchanges messages with others along the spanning tree, and eventually gets the lock. Then, it commits the arrival of the request by sending this commitment information to all the affected servers. All the servers that receive this commitment run the same EDF-CE (for an add/remove request of a periodic task) or recursive-push (for an on-demand job) algorithm to update their schedule tables.

This schedule update procedure can be best illustrated by the example of Figure 3.17. Suppose that the initial lock holder is $S_2$ as shown in the left-most tree. Also, assume that an on-demand job $J_1(S_5, S_4, Iperf, 10)$ arrives at $S_5$ at time $t_1$. Since $S_5$ is not the

![Figure 3.17: Distributed schedule update](image)

Figure 3.17: Distributed schedule update
lock-holder, it enqueues its ID $S_5$ in the $S_5$’s queue and sends a LOCK-REQUEST message to the neighbor $S_3$ pointed by its $dir$ variable. $S_3$ is not the lock-holder either and thus it enqueues the requester’s ID $S_5$ and sends a LOCK-REQUEST message to the neighbor $S_2$ pointed by its $dir$ variable. In the meantime, suppose that another request $J_2(S_1, S_3, Iperf, 10)$ arrives at $S_1$ at time $t_2$. Since $S_1$ is not the lock-holder, it enqueues its ID $S_1$ and sends a LOCK-REQUEST message to $S_2$ pointed by its $dir$ variable. When $S_1$’s LOCK-REQUEST reaches the lock-holder $S_2$, $S_2$’s queue is empty and it is not updating the schedule (not in the critical section), and thus it can immediately yield its lock to the requester $S_2$ by sending a LOCK-APPROVAL message to the requester $S_1$. It is no longer the lock-holder and set its $dir$ variable to $S_1$ (see the second tree).

When $S_1$ receives the LOCK-APPROVAL at $t_3$, it notices that the head of the queue is its own ID and thus can enter the critical section to commit $J_2$’s arrival. Since the $J_2$’s arrival needs to be viewed by all affected servers in a consistent way, $S_1$ adds the sufficient delay $\Delta t$ of commitment transmission to $t_3$ and considers $t_3 + \Delta t$ as the committed arrival time of $J_2$. Then, $S_1$ sends the commitment information ($J_2$ and $t_3 + \Delta t$) to the all affected servers, $S_2$ and $S_3$. Now, $S_1$, $S_2$, and $S_3$ can run the same recursive-push algorithm for inserting $J_2$ with the same committed arrival time of $t_3 + \Delta t$. When the LOCK-REQUEST message from $S_3$ arrives at $S_2$, $S_2$ is not the lock-holder and its $dir$ is pointing $S_1$. Thus, the LOCK-REQUEST is forwarded to $S_1$.

When the LOCK-REQUEST reaches $S_1$, it is the lock-holder but it is already in the critical section to commit $J_2$. Thus, $S_1$ enqueues the requester’s ID $S_2$. After that, $S_1$ leaves the critical section at $t_3 + \Delta t$. At this time, $S_1$ notices that the head of its local queue is $S_2$ and thus sends a LOCK-APPROVAL message to $S_2$ and sets its $dir$ toward $S_2$. $S_2$ and $S_3$ in turn forward the LOCK-APPROVAL and update their $dir$ variables according
to the head of their queues until the LOCK-APPROVAL reaches $S_5$. When $S_5$ receives the LOCK-APPROVAL at time $t_4$, it notices that the head of its queue is itself and thus can enter the critical section to commit the arrival of $J_1$. The commitment phase is the same as that of $J_2$. As a consequence, the concurrent arrivals of $J_1$ and $J_2$ are globally serialized in the order of $J_2$ and $J_1$ with the consistent commitment times of $t_3 + \Delta t$ and $t_4 + \Delta t$. Therefore, the schedule can be updated in a globally consistent way, assuring the conflict-free scheduling property.

For the complete and formal description of this distributed schedule update procedure, the readers are referred to [29]. The procedure inherits the proved properties of Raymond’s algorithm, such as minimal message exchange for assuring serializability, deadlock-freedom, no-starvation, and fault-tolerance.

**Performance Evaluation**

In order to study the overhead of the distributed implementation of the scheduling algorithms, we simulate both centralized and distributed implementations of the recursive push algorithm in a large scale network. For the network topology, we use the Waxman topology with 1000 nodes produced by the BRITE tool [93]. From the topology of 1000 nodes, we randomly select $N$ nodes as the measurement servers creating a measurement topology with $N$ measurement servers over the network topology with 1000 nodes. We consider three different $N$s: 100, 200, and 300. These choices represent NMIs with a reasonably large number of servers noting that the largest NMI deployment today, i.e., the NLANR AMP project [4] has around 150 measurement servers deployed all over the world. On top of the measurement topology, we use a synthetic task set with 100 offline periodic measurement tasks. The period $p_i$ and the execution time $e_i$ of each task $\tau_i$ are randomly generated
Figure 3.18: Response time comparison of the centralized and distributed implementations from [20 minutes, 200 minutes] and [1 minute, 10 minutes], respectively. Then, the execution times of all 100 tasks are scaled such that the total utilization $\sum_{i=1}^{100} e_i/p_i$ become 50%. Each task is assigned with randomly selected \( src \) and \( dst \) servers. With this offline periodic task set, we generate the offline schedule using the EDF-CE algorithm. Given the offline schedule, we simulate the random arrival of 1000 on-demand jobs with random execution times following the exponential distribution with the average 5 minutes. We conduct the simulation as we increase the average arrival rate from 10 jobs/hour to 150 jobs/hour following the Poisson distribution. Each on-demand job is assigned with randomly selected \( src \) and \( dst \) servers. In the following figures, we report the average of 100 simulation runs.

Figure 3.18 compares the average response times of on-demand jobs by the centralized and distributed implementations of the recursive-push algorithm. The centralized and distributed implementations show almost the same response times. This is because the total message passing delay to transfer the lock in the distributed implementation is at most
2 seconds even in a large scale measurement topology with 300 servers as shown in Figure 3.19. Also, such delay does not increase with the increase of the arrival rate. This is due to the message minimization capability of Raymond’s algorithm as the number of requests increases [29]. Another interesting observation in Figure 3.18 is that the response time is smaller when the number of servers is larger. This is because the on-demand job workload is scattered over a larger number of servers and hence the per-server workload is smaller.

### 3.5 Multi-domain Active Measurement Federations

This section discusses the inter-domain NMI federation issues and explains how our scheduling framework can be incorporated into a federation. For building an NMI federation, all the participating ISPs should agree on the following: (1) sharing each other’s measurement server topology, (2) bounding the amount of measurement traffic (i.e., the
MLA constraint $\psi$), (3) authenticated and secure access to measurement resources, and (4) sharing collected measurement data.

First, the measurement server topology of an ISP can be securely revealed only to other ISPs in the same federation using the agreed authentication and encryption methods as will be discussed later. Thus, every measurement server in the NMI federation can have the federation-wide view of the server topology and thus can determine the schedule of measurement tasks even if they span across multiple ISPs. Second, the agreed measurement traffic bound, MLA constraint $\psi$, can be enforced in our scheduling algorithms as explained in Section 3.2 and thus it can be complied across multiple ISPs. Third, for the authenticated and secure access to measurement resources across ISP borders, all ISPs can use a pre-agreed authentication and encryption techniques. For example, upon arrival of a new measurement request, they can use a centralized Kerberos [88] authentication server with Data Encryption Standard or triple DES. This can verify that the requesting domain belongs to the same NMI federation and also prevent intruders from eavesdropping the request for deciphering the authentication mechanism and impersonation as a member of the NMI federation. Finally, the collected measurement data can be shared by multiple ISPs as needed by distributed computing applications, using “Request/Response” schemas being developed by the Global Grid Forum’s Network Measurements Working Group [20]. An example of a request and response schema is shown in Figure 3.20

We envision that the growth of a global NMI federation mostly involves dealing with policy issues rather than technical issues. Since the application and ISP communities are realizing the importance of NMI federations for inter-domain measurements, we believe that all the policy issues will be overcome in the near future. Note that such efforts have
Figure 3.20: Request/response schema example to exchange inter-domain measurements

been already started by the communities such as Global Grid Forum, Internet2 in USA, and DANTE in Europe [20] [21].

3.6 Case Study I: Network Monitoring

In this Section, we assess the utility of our measurement scheduling algorithms in the ActiveMon framework [30] that we developed. ActiveMon is being used for routine network monitoring of several networks such as OSCnet, Ohio’s regional network, ADECnet, American Distance Education Consortium network, and Internet2, a national research network connecting several regional and national networks in the USA.

3.6.1 ActiveMon Framework

Figure 3.21 illustrates the basic components of ActiveMon. The active measurement toolkit shown in Table 3.1 is used to perform active measurements between a set of measurement servers (nodes) located at strategic points in the network being monitored. The initiated active measurements across the multiple measurement servers are orchestrated using our offline and online measurement scheduling algorithms. A central database (root) is used to collect and analyze the on-going and on-demand measurement data. The analysis
component of ActiveMon includes a statistical analysis package that processes measurement data for both visualization and alarm reporting. The generated alarms are sent via e-mail to alert appropriate network operations personnel.

Figure 3.21: ActiveMon framework components

To facilitate user-input to create measurement schedules, ActiveMon supports a “Scripting Language Interface” at a central regulator i.e., root server as shown in Figure 3.22. The scripting language interface provides a generic and automated way to input measurement specifications such as measurement server topology, periodic measurement tasks, and MLAs. These specifications are interpreted by the central regulator to construct schedule timetables for the measurement servers. The constructed schedule timetables are transferred to the corresponding servers to initiate the measurement jobs in a timely manner.
QoS Metric | Tool Name
--- | ---
Round-trip delay | Ping
One-way delay | OWAMP
Route changes | Traceroute
Per-hop bandwidth | Pathchar
Available bandwidth | Pathload
Bottleneck bandwidth | Pathrate
Transfer bandwidth | Iperf
VoIP MOS | H.323 Beacon

Table 3.1: ActiveMon measurement toolkit

Figure 3.22: Structure of measurement scheduling framework
3.6.2 Conflict-free Network Monitoring

We now verify the trustworthiness of our scheduling algorithms to avoid measurement conflicts. For this, we deploy ActiveMon on an Internet2 testbed that has five sites. Each of the sites is equipped with a measurement server as shown in Figure 3.23(a). To collect the actual measurement data, we run five periodic measurement tasks as shown Figure 3.23(b). The resulting task conflict graph is shown in Figure 3.23(c).

Figure 3.24 shows the H.323 Beacon MOS reports measured between Site-3 and Site-4 by task $\tau_2$. To compare EDF-CE and no-orchestration, we pick the same 12-hour time frames in two consecutive days. For the 12-hour time frame of the first day, we use no-orchestration method to run all five measurement tasks in Figure 3.23(b) and collect the MOS reports from $\tau_2$. For the 12-hour time frame of the second day, we use EDF-CE and collect the same reports. From these two experiments, we can observe that the proposed EDF-CE guarantees zero conflict while no-orchestration causes 50% instances of $\tau_2$ to overlap with other tasks. All the overlaps in the no-orchestration schedule are indeed conflicts since all the tools used in Figure 3.23(b) are CPU intensive and channel intensive. In terms of MOS accuracy, however, we are not sure which curve is better reflecting the reality of the network status, since we do not know the “true-real” network status. In order
to have a good representation of the reality of the network status between Site-3 and Site-4, we run only $\tau_2$ over a week long period. The results are shown in Figure 3.25. From the figure, we can affirm that MOS fluctuation between 4.31 and 4.42 is natural in reality between Site-3 and Site-4. The MOS values in Figure 3.24 collected using EDF-CE well match the representation of the reality in Figure 3.25. In contrast, the MOS reports by the no-orchestration method in Figure 3.24 show much larger fluctuation, which seems abnormal comparing with Figure 3.25. We can conclude that these abnormal fluctuations are due to 50\% instances of $\tau_2$ conflicting with other tasks.

Figure 3.24: H.323 Beacon MOS measurements between Site-3 and Site-4

Figure 3.26 shows Iperf packet jitter reports measured between Site-3 and Site-2 by task $\tau_1$. We can clearly distinguish the reported jitter range by EDF-CE from that by no-orchestration. The large absolute jitter values and bigger fluctuation in the no-orchestration case is due to conflicts of $\tau_1$ with others in the no-orchestration schedule.
Figure 3.25: H.323 Beacon MOS measurements over a week period between Site-3 and Site-4

Figure 3.26: Iperf jitter measurements between Site-3 and Site-2
Figure 3.27 shows the Pathload Bandwidth reports measured between Site-3 and Site-5 by task \( \tau_4 \). The fluctuations of the bandwidth reports by EDF-CE are in the range of 88 Mbps to 98 Mbps. However, the fluctuations by no-orchestration in more than 50% of the time fluctuate outside the baseline range with several zero bandwidth reports. These zero bandwidth reports are due to the “interrupt coalescence” behavior [94], which is triggered when Pathload is concurrently executed with other tools in the no-orchestration method. Such wide fluctuations and several zero bandwidth reports are due to conflicts of \( \tau_4 \) in the no-orchestration schedule. The interrupt coalescence behavior can be explained as follows: When the CPU of a measurement server is fully consumed with processing network interrupts instead of processing the data contained in the received packets, the Network Interface Card (NIC) of the measurement server causes interrupt coalescence by generating a single interrupt for multiple packets sent or received in a short time interval. This interrupt coalescence behavior alters the “dispersion” (or inter-arrival time spacing) of packets.
and consequently affects the estimation technique used in Pathload that relies on the inter-arrival time spacing information to accurately estimate network bandwidth. Since EDF-CE measurements are conflict-free, they are not affected by interrupt coalescence.

From the above Internet2 WAN testbed results, we can justify the importance of measurement orchestration that avoids measurement conflicts while correctly estimating network status.

### 3.6.3 Network Performance Anomaly Detection

We now explain how our measurement scheduling algorithms implemented in ActiveMon enabled detection of notable network performance anomalies on the Internet. For this purpose, we deployed ActiveMon on an unique measurement testbed spanning three hierarchically different backbone network paths: campus, regional and national paths. The campus path refers to a network path between a lab at The Ohio State University (OSUL) and the OARnet border router for OSU campus (OSUB). On the campus path, only OSU campus backbone network routers were present in between. The regional path refers to a network path between OARnet border routers for OSU and University of Cincinnati (UOCB). On the regional path, only OARnet regional network backbone routers were present in between. The national path refers to a network path between OSUB and a North Carolina State University border router (NCSB) owned by NCNI. On the national path, only backbone routers of two regional ISPs (OARnet and NCNI) separated by a national ISP (Abilene) were present in between.
The most common anomalies observed while monitoring network paths are those caused by route changes. Route changes are attributed to “route flaps” caused by suboptimal routing protocol behavior, network infrastructure failures, re-configuration of networks or load-balancing strategies used by ISPs to improve network performance. Figure 3.28 shows a visualization created by ActiveMon to detect route changes obtained by Traceroute data analysis along the campus path. In the six month period shown, route changes occurred four times. During the same period, we detected only two route changes in the regional path and no route changes occurred in the national path. We determined that the reason for the high frequency of route changes in the campus path was due to the various network management activities associated with the transitioning of the campus traffic from an old ATM-based network to an IP-based network. The route numbers and hop numbers shown in Figure 3.28 correspond to distinct route signatures and host IP addresses stored in the ActiveMon’s root database tables.

Figure 3.29 shows the round-trip delay measurements provided by Ping and OWAMP during the six month monitoring period for the same campus backbone network path. For OWAMP which measures high-precision one-way delay, we calculate the round-trip delay using the sum of the bidirectional one-way delay measurements along the path. We can notice that the variations in trends of the Ping and OWAMP measurements respond in a similar fashion. Another interesting observation is that the delay variations directly correspond to the route changes shown in Figure 3.28. We can also observe sudden short-lived dips and peaks in the measurements due to miscellaneous temporal network dynamics in the network path. In the same time and path context, we analyze the jitter measurements provided by Iperf using Figure 3.30. We can notice that the variations in trends of the jitter measurements occurred only for the first route change along the campus path. The
Figure 3.28: Route changes detected along campus backbone network path
other route changes did not affect the jitter along the path noticeably. We can also notice that on December 15th, a random network event, which does not correspond to a route change, caused an increased variance in the jitter measurements. Further, we can notice jitter anomaly characteristics such as spikes (or temporal anomalies) and short and long burst patterns (or spatial anomalies).

Another interesting network performance anomaly was observed in the OWAMP measurements. We refer to this anomaly as the “ocean-wave anomaly”, which is shown in Figure 3.31. The cause of this anomaly was identified as follows: OWAMP tool relies on the NTP clock offset to report one-way delays at the microsecond precision-level. Due to NTP instability in the clocks at UOCB, an interesting pattern of one-ways delays occurred in the OWAMP measurements involving UOCB. Periodically as the NTP estimates skewed off, NTP compensated for the extreme skew and gave rise to the ocean-wave anomaly in the OWAMP measurements.
Figure 3.30: Jitter anomalies detected along campus backbone network path

Figure 3.31: OWAMP “ocean-wave anomaly” detected on regional backbone network path
Yet another interesting network performance anomaly was observed in the pathload bandwidth measurements. The anomaly is shown in Figure 3.32, which occurred when a rate-limit was accidentally misconfigured at the OSUB attached router. We can notice a significant drop in the available bandwidth measurements. The average available bandwidth changed from 80.37 Mbps to 59.85 Mbps on the 100Mbps link. After the misconfiguration was reported and corrected, there resulted an increase in the average available bandwidth to about 82.53 Mbps.

![Bandwidth anomaly detected on regional backbone network path](image)

Figure 3.32: Bandwidth anomaly detected on regional backbone network path

**Relative Performance Evaluation of Network Paths**

Besides using the measurement data for detecting anomalies, we also use the data to assess the relative performance of the campus, regional and national network paths. For this, we choose the statistical coefficient of variation ($\rho$) as a relative performance measure.
The \( \rho \) is calculated as a percentage as follows:

\[
\rho = \frac{S}{\bar{X}} \times 100
\]

where -

\[
S = \sqrt{\frac{\sum (x_i - \bar{X})^2}{N}}; \bar{X} = \frac{\sum x_i}{N}
\]

where, \( x_i \) is the \( i^{th} \) observation, \( N \) is the number of non-missing observations, \( \bar{X} \) is the mean and \( S \) is the standard deviation. The \( \rho \) values obtained from the active measurements for the campus, regional and national backbone network paths are shown in Table 3.2. Since lower variability is implied by a lower value of \( \rho \), we can conclude that regional paths have the overall best network performance. Thus, they may cause the least QoE degradation for end-users. Whereas, both campus and national paths show higher values of \( \rho \) i.e., variability. Consequently, they are more probable to cause end-to-end performance bottlenecks that degrade end-user QoE. Such relative performance comparisons can be extended using ActiveMon for monitoring network-wide paths. From the obtained measurements data, listings such as the “top-ten best and worst performing network paths” can be generated routinely. Such listings are a highly desired feature when monitoring large network backbones [21].

<table>
<thead>
<tr>
<th>Tool Characteristic</th>
<th>Campus</th>
<th>Regional</th>
<th>National</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pathrate Max. Bandwidth (Mbps)</td>
<td>24.23</td>
<td>5.5</td>
<td>6.74</td>
</tr>
<tr>
<td>Iperf Jitter (ms)</td>
<td>745.95</td>
<td>45.1</td>
<td>499.89</td>
</tr>
<tr>
<td>Iperf Loss (%)</td>
<td>517.63</td>
<td>62.48</td>
<td>127.43</td>
</tr>
<tr>
<td>H.323 Beacon MOS</td>
<td>18.48</td>
<td>0.03</td>
<td>9.63</td>
</tr>
<tr>
<td>OWAMP Delay (ms)</td>
<td>52.87</td>
<td>10.64</td>
<td>13.47</td>
</tr>
<tr>
<td>Ping Delay (ms)</td>
<td>58.52</td>
<td>5.65</td>
<td>24.4</td>
</tr>
</tbody>
</table>

Table 3.2: Relative performance of the academic backbone network paths
From the anomaly observations and relative performance comparison study explained above, we can thus see that our scheduling algorithms are pertinent for routine network monitoring on the Internet.

3.7 Case Study II: Network Weather Forecasting

We now apply our measurement scheduling algorithms to the widely-used Network Weather Service (NWS) [14] framework that can provide network performance forecasts.

3.7.1 Network Weather Service (NWS) Framework

NWS uses periodically measured network status and forecasts the network performance. Due to its ability of forecasting network performance, NWS has been adopted by a number of networked job schedulers such as AppLes [90], Legion [91], Globus/Nexus [92]. In this application, one challenge is the gap between the original measurement time requirement of NWS and the actual temporal behavior of our scheduling algorithms. More specifically, NWS periodically issues measurement requests expecting a periodic sampling of network status. However, the scheduler cannot serve the requests exactly at the desired times due to resource conflicts with other measurement requests. As such, any scheduler that tries to avoid conflicts, inevitably creates a jitter in the inter-sampling times of network status. This section presents a simple method for compensating the inter-sampling jitter.

NWS relies on continuous and periodic sampling or Pure Periodic Sampling (PPS) of network status. It uses the periodically sampled network status data to maintain the history of network performance, which in turn is used to generate on-going and dynamic network performance forecasts. The forecast time window is the same as the sampling period.

However, it is not always possible for the measurement scheduling algorithm to provide pure-periodic network status data, especially when multiple measurement tasks are
running. This can be explained by Figure 3.33 that shows an example conflict-free schedule of two periodic measurement tasks that have a conflict relation. We can note that the task $\tau_2$ is scheduled pure-periodically with constant inter-sampling times. However, the inter-sampling times of task $\tau_1$ vary for every instance. To avoid conflict of multiple concurrent tasks, the actual scheduling time points are inevitably deviated from the periodic release time points by any conflict-free scheduling algorithm. Our EDF-CE also produces such inter-sampling time jitter since it is designed to guarantee the periodic deadlines and not pure periodic execution of jobs. In fact, with our EDF-CE, the inter-sampling jitter of task $\tau_1$ can vary from $e_i$ (when a job instance is scheduled just before its deadline and the next one is scheduled at the release time) to $2p_i - e_i$ (when a job instance is scheduled at the release time and the next one is scheduled just before the deadline).

### 3.7.2 Transformed Periodic Sampling

Although our EDF-CE causes inter-sampling jitter between two consecutive jobs, it bounds the jitter by meeting the end-of-period deadlines. Therefore, it can still be used by NWS with simple interpolations of collected network status data. The interpolation involves transforming the actual measured data to pure-periodic data using piece-wise linear interpolation. To explain this, let us consider Figure 3.34. In the figure, $(t_{i-1}, y_{i-1})$, $(t_i, y_i)$, and $(t_{i+1}, y_{i+1})$...
(t_i, y_i), and (t_{i+1}, y_{i+1}) show the sequence of the \((i - 1)\)-th, \(i\)-th, and \((i + 1)\)-th actual periodic sampling (APS) whose inter-sampling time is not always the same as the period \(p\). To transform the APS sequence to a pure-periodic sampling sequence, which we call Transformed Periodic Sampling (TPS), we can draw piece-wise lines between pairs of two APS points. For example, we can draw a line between \((t_{i-1}, y_{i-1})\) and \((t_i, y_i)\) as shown in Figure 3.34. With this line, we can estimate the measurement value \(\hat{y}_i\) at the pure-periodic \(i\)-th sampling time, i.e., \((i - 1) \cdot p\). Specifically, \(\hat{y}_i\) is given as follows:

\[
\hat{y}_i = y_i + \frac{y_{i+1} - y_i}{t_{i+1} - t_i} (i - 1) \cdot p - t_i.
\]

Thus, the APS data \((t_i, y_i)\) with inter-sampling jitter can be transformed to the pure-periodic TPS data \(((i - 1) \cdot p, \hat{y}_i)\) as shown in Figure 3.34. Similarly, \((t_{i+1}, y_{i+1})\) can be transformed to \((i \cdot p, \hat{y}_{i+1})\). Now, the NWS can use the TPS data rather than the original measured data to provide the network performance forecast.
3.7.3 Forecasting Accuracy Analysis

We now show that our EDF-CE can work well in combination with NWS to produce accurate forecasts, when it uses the simple interpolation method explained in the previous sub-section. For this purpose, we use actual trace data obtained from NWS measurements for a path traversing a T1 connection with a total bandwidth 1.5 Mbps [89]. The trace data corresponds to “hourly” samples of available bandwidth on the T1 line over a two-day period. We assume that the trace data reflects the “actual” network performance trend on the path. Using this trace data, we generate two sample sequences, one representing the ideal pure-periodic sampling (PPS) with a period of 2 hours, and another one that corresponds to actual sampling (APS) by EDF-CE. The 2-hour-based PPS is obtained by considering every other sample in the trace data assuming that no other monitoring tasks are present. This 2-hour-based PPS is the ideal measurement sequence expected by NWS for 2-hour look-ahead forecasts. The actual measurement sequence, APS, however, inevitably deviates from the PPS due to scheduling of conflicting tasks. To model the APS, we simulate the EDF-CE with four measurement tasks in Figure 3.35. Note that task $\tau_1$ serves NWS by providing 2-hour based sampling of available bandwidth. This simulation provides the inter-sampling time distribution of $\tau_1$, which is used to select the samples from the trace data.
Figure 3.36: Comparison of Forecasts of PPS and APS

data. The selected samples approximately represent the actual sequence of samples obtained by $\tau_1$ as scheduled by EDF-CE in the presence of three other tasks$^4$.

Figure 3.36 shows the reality of the available bandwidth (i.e., one hour based trace data), the NWS forecasted bandwidth using \textit{PPS}, and the same using \textit{APS}. The NWS forecasting using the ideal \textit{PPS} closely match the actual trend. However, the NWS forecasting using \textit{APS} has non-negligible differences from the reality at several instances. This is because of inter-sampling time jitter caused by EDF-CE. This problem can be fixed by a simple transformation of sampled data as described in the previous sub-section. Figure 3.37 shows that the NWS forecasting using the transformed samples denoted by \textit{TPS} is very similar to the ideal forecasting by \textit{PPS}.

$^4$Due to the one hour based granularity of the original trace data, the sequence of selected samples is only an approximation with quantization errors up to one hour. However, it is acceptable in terms of showing how the aforementioned simple interpolation can resolve the inter-sampling time jitter caused by EDF-CE, which ranges from 0 to 4 hours.
Instead of visually observing the closeness of the forecasting curves, a more suitable quantitative method to compare the NWS forecasting performance obtained by using $PPS$, $APS$ and $TPS$ is to use the Mean Square Error (MSE) fitness metric. MSE quantifies the lowest prediction error. Hence, lower the MSE values, better is the forecasting performance. MSE is calculated as follows:

$$MSE_f(t) = \frac{1}{t+1} \sum_{i=0}^{t} (err_f(i))^2$$  \hspace{1cm} (3.1)

where,

$$err_f(t) = value(t) - prediction_f(t-1)$$  \hspace{1cm} (3.2)

The $value(t)$ is the measured bandwidth at time $t$, $prediction_f(t-1)$ is its prediction done by NWS at time $(t-1)$ and $err_f(t)$ is the prediction error. Figure 3.38 shows the MSE values produced by the NWS forecasting models while forecasting the network bandwidth values between time points $A$ and $A'$ shown in Figures 3.36 and 3.37. The effect of high-to-low
variation of bandwidth data between the $A$ and $A'$ time points can be seen in the increasing and then decreasing trend of the MSE values as the bandwidth data stabilizes. Amongst the MSE values obtained using the $PPS$, $APS$ and $TPS$ datasets, we can note that the MSE values are relatively smaller for the forecasting with $PPS$ data. By this, we can conclude that $PPS$ data is best suited for forecasting purposes. Also, by comparing the relatively lower MSE values obtained using the $TPS$ data with those obtained using the $APS$ data, we can conclude that the transforming process considerably improves the overall forecasting performance of NWS by reducing the MSE by almost half. Further, the improvement when using $TPS$ data is comparable to the forecasting performance achievable if using the $PPS$ data.
3.8 Summary

In this chapter, we presented our offline (heuristic bin packing, EDF-CE) and online (recursive-push) measurement scheduling algorithms that can handle multi-domain ongoing and on-demand measurements, respectively. We showed that by using our Concurrent Execution (CE) principle, measurement schedulability can be increased by more than 100% till upto 300% depending on the conflict factor. At the same time, we showed how measurement regulation was possible using the Measurement Level Agreement (MLA) specification. We also demonstrated how our recursive push algorithm was able to significantly reduce the average response time of on-demand jobs by 50%, without disrupting the offline EDF-CE generated schedules. Further, we explained how our scheduling algorithms could be implemented without a central regulator i.e., in a distributed manner. For this, we used the Raymond’s algorithm and showed how to achieve negligible implementation overhead for the distributed schedule management. Further, we showed that with minimal message passing, we could assure serializability, deadlock-freedom, no-starvation and fault-tolerance while updating the distributed schedules. In the last part of this chapter, we presented two case studies that showed the utility of our measurement scheduling algorithms. The first case study showed the utility in the ActiveMon framework, which is used for routine network monitoring. The second case study showed the utility in the Network Weather Service (NWS) framework, which is used for network weather forecasting.
CHAPTER 4

VPERP TOOL FOR VVOIP QOE ESTIMATION

In this chapter, we describe our Vperf tool for measuring streaming and interaction VVoIP QoE over the Internet. In Section 4.1, we consider the combined impact of all the QoS metrics and map their impact in terms of end-user QoE. In this context, we show how doing so would involve an unreasonable amount of testing with human subjects and a complex test setup. To address this, we explain our test case reduction strategies in Section 4.1.1 and the closed-network testing with human subjects in Section 4.1.2. In Section 4.1.3, we analyze the data obtained from the closed-network testing to derive the GAP-Model’s closed-form expressions. In Section 4.1.4, we describe the GAP-Model characteristics and illustrate how it considers combined effects of QoS metrics for estimating VVoIP QoE. In Section 4.1.5, we describe how we implement the GAP-Model in the Vperf tool. In Section 4.1.6, we validate the GAP-Model with additional human subject tests. We also compare the GAP-Model’s online estimates of VVoIP QoE with the J.144’s offline estimates i.e., we compare GAP-Model MOS rankings with PSNR-mapped-to-MOS rankings. Finally, in Section 4.2, we describe how we quantify VVoIP interaction QoE in terms of the interaction difficulties experienced by end-users using the Multi-Activity Packet Trains (MAPTs) methodology. In Sections 4.2.1 and 4.2.2, we explain the video activity levels in VVoIP sessions and show how they correspond to the participant interaction patterns. In Sections
4.2.3, 4.2.4 to 4.2.5, we explain the packet-train characteristics i.e., instantaneous video packet sizes and encoding rates, respectively for emulating the video activity levels and participant interaction patterns under different network fault event conditions. In Section 4.2.6, we describe our MAPTs implementation in the Vperf tool. In Section 4.2.7, we evaluate the MAPTs measurements in a network testbed that features a wide variety of network fault events.

4.1 VVoIP Streaming QoE Estimation

Unlike earlier studies relating to QoE degradation which considered isolated effects of individual network factors such as loss [63] and bandwidth [102], we consider the combined effects of the different levels of $b_{\text{net}}$, $d_{\text{net}}$, $l_{\text{net}}$, and $j_{\text{net}}$, each within a GAP performance level. Although such a consideration reflects the reality of the network conditions seen on the Internet, modeling $q_{\text{mos}}$ as a function of the four network factors in three different levels leads to a large number of test cases (i.e., $3^4 = 81$ test cases) per human subject. The test cases can be ordered based on increasing network condition severity and listed as [<GGGG>, <GGGA>, <GGAG>, ..., <APPP>, <PPPP>], where each test case is defined by a particular sequence of the network factor levels $<b_{\text{net}}, d_{\text{net}}, l_{\text{net}}, j_{\text{net}}>$.

For example, the <GGGG> test case corresponds to the network condition where $b_{\text{net}}$, $d_{\text{net}}$, $l_{\text{net}}$ and $j_{\text{net}}$ are in their Good grade levels. Administering all the 81 test cases per human subject is an expensive process and also involves long hours of testing that is burdensome and exhaustive to the human subject. Consequently, the perceptual QoE rankings provided by the human subject may be highly error-prone.

To overcome this, we present a novel closed-network test methodology in the following sub-section that significantly reduces the number of test cases. In addition, it also reduces
the testing time for human subjects for providing rankings without compromising the rankings data required for adequate model coverage. We note that our test methodology can be generalized for any voice (e.g. G.711, G.722, G.723) and video codec (e.g. MPEG-x, H.26x). For simplicity, we focus our testing to only the most commonly used codecs i.e., G.722 voice codec and the H.263 video codec. These codecs are used as the primary codecs in most videoconferencing end-points and on video file sharing sites such as MySpace and Google Video.

4.1.1 Test Case Reduction Strategies

To reduce the number of test cases per human subject for providing rankings without compromising the rankings data required for adequate model coverage, we use two strategies: (i) reduction based on network condition infeasibility and (ii) reduction based on human subjects’ ranking inference.

Reduction Based on Network Condition Infeasibility

For this strategy, we perform a network emulator qualification study to identify any practically infeasible network conditions i.e., test cases that do not exist in reality. The NISTnet WAN emulator is connected in between two isolated LANs, each having a measurement server with the Iperf tool [11] installed. Different network conditions are emulated with one factor as the control and the other factors as the response. For example, if we use $b_{net}$ as the control, then the responses of the other three factors $d_{net}$, $h_{net}$ and $j_{net}$ are measured and so on. All measurements are from Iperf for 768 Kbps UDP traffic streams transferred between the two LANs via NISTnet.

5Majority of today’s videoconferencing end-points use the H.263 video codec and a small fraction of the latest end-points support the H.264 video codec, which is an enhanced version of the H.263 codec targeted mainly for improved codec performance at low bit rates.
Figures 4.1 and 4.2 show the Iperf measurement results that indicate the infeasible network conditions. The results are averaged over 20 measurement runs of Iperf for each network condition configuration on NISTnet. From Figure 4.1 we can see that there cannot be a network condition that has Good $j_{net}$ and Poor $b_{net}$ simultaneously. Hence, $< P \ast G >$ (i.e., $= 1 \times 3 \times 3 \times 1 = 9$) test cases cannot be emulated in reality. Note here that we use our previously defined network condition notation $< b_{net} \, d_{net} \, l_{net} \, j_{net} >$ and we assume $\ast$ can be substituted with either one of the GAP grades. Similarly, from Figure 4.1 we can see that there cannot be network conditions that have Good/Acceptable $l_{net}$ and Poor $b_{net}$ simultaneously. Hence, $< P \ast G \ast >$, $< P \ast A \ast >$, $< A \ast G \ast >$ and $< A \ast A \ast >$ (i.e., $9 \times 4 = 36$) test cases do not exist in reality. If we combine all the infeasible network conditions, we get a list of 45 test cases. By eliminating duplicates in this list, we end up with a total of 39 test cases. Hence, we can reduce the number of test cases to 42 (i.e., $39$ subtracted from $81$) per human subject for adequate model coverage.
Reduction Based on Human Subjects’ Ranking Inference

In this subsection, we explain another strategy to further reduce the number of test cases per human subject for providing rankings without compromising the data required for adequate model coverage. The basic idea of this strategy is to eliminate more severe test cases during the testing based on the Poor rankings given by human subjects for relatively less severe test cases. For example, if a human subject ranked test case $<GPPP>$ with an extremely Poor $q_{\text{mos}}$ ranking ($< 2$), it can be inferred that more severe test cases $<APPP>$ and $<PPPP>$ presented to the human subject will also result in extremely Poor $q_{\text{mos}}$. Hence, we do not administer the $<APPP>$ and $<PPPP>$ test cases to the human subject during testing but assign the same Poor $q_{\text{mos}}$ ranking obtained for $<GPPP>$ test case to the $<APPP>$ and $<PPPP>$ test cases in the human subject’s final testing results.
To implement the above test case reduction strategy, we can either present the test cases with increasing severity (\(< GGGG >\) to \(< PPPP >\)) or decreasing severity (\(< PPPP >\) to \(< GGGG >\)) to each human subject. We choose the increasing test case severity option because human subject testing regulations explicitly discourage exposing human subjects to extreme/abnormal conditions. Such conditions would occur if we chose to present the test cases with decreasing severity.

Further, the test case reduction strategy can be implemented by increasing the test case severity order in two ways: (i) Vertical-First (VF) or (ii) Horizontal-First (HF) - shown in Figure 4.3(a) and (b), respectively. Using VF ordering, after \(< GGGA >\), the next severe condition in the test case list is chosen as \(< GGGP >\) where the severity is increased vertically (note that \(< GGGA >\), \(< GGAG >\) and \(< GAGG >\) are equivalent severe conditions); whereas, using HF ordering, the next severe condition is chosen as \(< GGAA >\) where the severity is increased horizontally. In the event that \(< GGAA >\) test case receives an extremely Poor \(q_{mos}\) ranking (< 2) by a human subject, 36 (= 3x3x2x2) test cases get eliminated using the inference strategy. Alternately, if \(< GGGP >\) test case receives...
an extremely Poor $q_{mos}$ ranking, only 27 ($= 3 \times 3 \times 3 \times 1$) test cases get eliminated. Hence, using the VF ordering, relatively lesser test cases are eliminated when an extremely Poor $q_{mos}$ ranking occurs. Although HF ordering reduces the testing time compared to the VF ordering, we choose the VF ordering in the human subject testing because it produces more data points and thus relatively better model coverage.

4.1.2 Closed-network Testing

In this sub-section, we describe our closed-network testing with actual human subjects.

Test Environment Setup

There are several challenges in setting up a test environment that provides repeatability in obtaining QoE rankings from several human subjects. First, the environment factors need to be completely controlled, i.e., there cannot be extraneous factors such as network cross traffic and loud noises that affect the testing conditions. Second, the testing must be automated as much as possible to minimize human errors in test administration. However, complete automation is not recommended by ITU-T to record human subject rankings. For example, a human subject should be able rest between test cases if he/she desires and eventually indicate his/her readiness to the test administrator to initiate a test case. Also, the human subject must be able to revise an accidentally submitted subjective ranking for a test case before the next test case is initiated. Finally, the human subject’s graphical user interface to enter QoE rankings must be intuitive and easy to use. This eases the initial human subject training before the testing and the subjective rankings submission process during the testing. Additional key guidelines are provided in ITU-T P.911 [97] and ITU-T recommendations P.920 [98] for non-interactive and interactive multimedia QoE assessment tests, respectively.
Figure 4.4: Test environment setup for the closed-network testing

Figure 4.5: Screenshot of chat application with quality assessment slider

Figure 4.4 shows the test environment we setup for the closed-network testing that incorporated the above key considerations. An isolated LAN testbed was used with no network cross-traffic whatsoever. The test station for the human subject was setup in a quiet room with sufficient light and ventilation. The test station corresponds to a PC that runs a chat application (shown in Figure 4.5) and a videoconferencing end-point connected
to a display terminal. The chat application uses the “quality assessment slider” methodology recommended by [99] for recording human subject rankings. The chat application allowed the human subject to: (a) communicate his/her test readiness using the “Begin Test” button, (b) indicate completion of his response during interactive test clips using the “Response Complete” button, and (c) submit subjective rankings using the “MOS Ranking” field at the end of each test case - to the test administrator present in a separate room. The videoconferencing end-point was used to view the streaming and interactive test clips.

The test administrator end was equipped with a PC that ran an in-band chat application that allowed the human subject to experience interaction delay conditions that influence rankings of interactive clip test cases. The test administrator end was also equipped with a videoconferencing end-point connected to a display terminal as well as a test clips source. The test administrator had controls to the test clips source as well as the NISTnet through a control software installed on the chat application PC. The control software guided the test administrator through the different sequential steps involved in the testing and automatically performed core actions to control the clips source and the NISTnet configurations. Using the control software, the test administrator could initiate a test case upon arrival of a readiness message from a human subject to begin a test case and also seamlessly control interactive video clips based on the response completion messages. Further, the test administrator could use the control software to simultaneously configure the appropriate network conditions on the NISTnet before initiating a test case. The pseudo-codes of the control software on the test administrator PC for the streaming and interactive tests are as follows:
Procedure 3 Pseudo-code of the control software for streaming tests

1: **Input:** 42 test cases list for streaming tests
2: **Output:** Subjective MOS rankings for the 42 test cases
3: **begin procedure**
4: **Step-1: Initialize Test**
5: Prepare \( j = 1 \ldots 42 \) test cases list with increasing network condition severity
6: Initialize NISTnet WAN emulator by loading the kernel module and flushing inbound/outbound pipes
7: Initialize playlist in streaming clip source
8: Play streaming “baseline clip” for human subject no-impairment stimulus reference
9: **Step-2: Begin Test**
10: Enter \( i^{th} \) human subject ID
11: **loop** for \( j \) test cases: if(Check for receipt of \( j^{th} \) “Begin Test” message)
12: Flush NISTnet WAN emulator’s inbound/outbound pipes
13: Configure the network condition commands on NISTnet WAN emulator for \( j^{th} \) test case
14: Play streaming test clip from the clips source
15: if (Check for receipt of “MOS Ranking” message)
16: Reset streaming test clip in the clips source
17: Save \( i^{th} \) human subject’s \( j^{th} \) streaming MOS ranking to database
18: if (\( i^{th} \) human subject’s \( j^{th} \) streaming MOS ranking < 2)
19: Remove \( k \) corresponding higher severity test cases from test case list
20: for each \( k \)
21: Assign \( i^{th} \) human subject’s \( j^{th} \) streaming MOS ranking
22: end for
23: end if
24: Increment \( j \)
25: end if
26: **end loop**
27: **Step-3: End Test**
28: Shutdown the NISTnet WAN emulator by unloading the kernel module
29: Close playlist in streaming clip source
30: **End procedure**
Procedure 4 Pseudo-code of the control software for interactive tests

1: **Input:** 42 test cases list for interactive tests
2: **Output:** Subjective MOS rankings for the 42 test cases
3: **begin procedure**
4: **Step-1: Initialize Test**
5: Prepare $j=1\ldots42$ test cases list with increasing network condition severity
6: Initialize NISTnet WAN emulator by loading the kernel module and flushing inbound/outbound pipes
7: Initialize playlist in interactive clip source
8: Play interactive “baseline clip” for human subject no-impairment stimulus reference
9: **Step-2: Begin Test**
10: Enter $i^{th}$ human subject ID
11: **loop for** $j$ test cases: if(Check for receipt of $j^{th}$ “Begin Test” message)
12: Flush NISTnet WAN emulator’s inbound/outbound pipes
13: Configure the network condition commands on NISTnet WAN emulator for $j^{th}$ test case
14: Play interactive test clip from clips source
15: Pause interactive test clip at the time point where human subject response is desired
16: **if** (Check for receipt of “Response Complete” message)
17: Continue playing the interactive test clip from the clips source
18: **end if**
19: **if** (Check for receipt of “MOS Ranking” message)
20: Reset interactive test clip in the clips source
21: Save $i^{th}$ human subject’s $j^{th}$ interactive MOS ranking to database
22: **if** ($i^{th}$ human subject’s $j^{th}$ interactive MOS ranking $<2$)
23: Remove $k$ corresponding higher severity test cases from test case list
24: **for** each $k$
25: Assign $i^{th}$ human subject’s $j^{th}$ interactive MOS ranking
26: **end for**
27: **end if**
28: Increment $j$
29: **end if**
30: **end loop**
31: **Step-3: End Test**
32: Shutdown the NISTnet WAN emulator by unloading the kernel module
33: Close playlist in interactive clip source
34: **End procedure**
**Human Subject Selection**

Human subject selection was performed in accordance with the Ohio State University’s Institutional Review Board (IRB) guidelines for research involving human subjects. The guidelines insist that human subjects should voluntarily participate in the testing and must provide written consent. Further, the human subjects must be informed prior about the purpose, procedure, potential risks, expected duration, confidentiality protection, legal rights and possible benefits of the testing. Regarding the number of human subjects for testing, ITU-T recommends 4 as a minimum total for statistical soundness [86].

To obtain a broad range of subjective quality rankings from our testing, we selected a total of 21 human subjects evenly distributed across three categories (7 human subjects per category): (i) Expert user, (ii) General user, and (iii) Novice user. An Expert user is one who has considerable business-quality videoconferencing experience due to regular usage and has in-depth system understanding. A General user is one who has moderate experience due to occasional usage and has basic system understanding. A Novice user is one who has little prior business-quality videoconferencing experience but has basic system understanding. Such a categorization allowed collection of subjective quality rankings that reflect the perceptual quality idiosyncrasies dependent on a user’s experience level with VVoIP technology. For example, an Expert user considers lack of lip-synchronization as a more severe impairment than audio drop-outs, which happens to be the most severe impairment for a Novice user - while penalizing MOS rankings. We remark that the human subjects belonged to diverse demographics as well and hence our rankings data covers opinions that may be influenced by age, race, and other such human population characteristics.
Video Clips

For test repeatability, each human subject was exposed to two sets of clips for which, he/she provided $q_{mos}$ rankings. The first set corresponded to a streaming video clip Streaming-Kelly and the second set corresponded to an interactive video clip Interactive-Kelly, both played for the different network conditions specified in the test case list. These two video clips were encoded at 30 frames per second in CIF format (352 lines x 288 pixels) and are available at [100]. The duration of each clip was approximately 120 seconds and hence provided each human subject with enough time to assess perceptual quality. Our human subject training method to rank the video clips is based on the “Double Stimulus Impairment Scale Method” described in the ITU-R BT.500-10 recommendation [101]. In this method, baseline clips of the streaming and interactive clips are played to the human subject before commencement of the test cases. These clips do not have any impairment due to network conditions i.e., $q_{mos}$ ranking for these clips is 5. The human subjects are advised to rank their subjective perceptual quality for the test cases relative to the baseline subjective perceptual quality.

Test Cases Execution

Before commencement of the testing, the training time per human subject averaged about 15 minutes. Each set of test cases per human subject for the streaming as well as interactive video clips lasted approximately 45 minutes. Such a reasonable testing time was achieved due to:

(a) our first test case reduction strategy described in Section 4.2.1 that reduced the 81 possible test cases to a worst case testing of 42 test cases, and
our second test case reduction strategy described in Section 4.2.1 that further reduced the number of test cases during the testing based on inference from the subjective rankings.

For emulating the network condition as specified by a test case, the network factors had to be configured on NISTnet to any values within their corresponding GAP performance levels shown in Table 2.6.1. We configured values in the performance levels for the network factors as shown in Table 4.1.2. For example, for the \(< GGGG >\) test case, the NISTnet configuration was \(< b_{net} = 960 Kbps; d_{net} = 80 ms; l_{net} = 0.25\%; j_{net} = 10 ms >\). The reason for choosing these values was that the instantaneous values for a particular network condition configuration vary around the configured value (although the average of all the instantaneous values over time is approximately equal to the configured value). Hence, choosing the values shown in Table 4.1.2 enabled sustaining the instantaneous network conditions to be within the desired performance levels for the test case execution duration.

<table>
<thead>
<tr>
<th>Network Factor</th>
<th>Good Level</th>
<th>Acceptable Level</th>
<th>Poor Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b_{net})</td>
<td>960 Kbps</td>
<td>768 Kbps</td>
<td>400 Kbps</td>
</tr>
<tr>
<td>(d_{net})</td>
<td>80 ms</td>
<td>280 ms</td>
<td>600 ms</td>
</tr>
<tr>
<td>(l_{net})</td>
<td>0.25 %</td>
<td>1 %</td>
<td>2 %</td>
</tr>
<tr>
<td>(j_{net})</td>
<td>10 ms</td>
<td>35 ms</td>
<td>75 ms</td>
</tr>
</tbody>
</table>

Table 4.1: Values of network factors within GAP performance levels for NISTnet configuration
4.1.3 Closed-form Expressions

In this subsection, we derive the GAP-Model’s closed-form expressions using $q_{mos}$ rankings obtained from the human subjects during the closed-network testing. As stated earlier, subjective testing for obtaining $q_{mos}$ rankings from human subjects is expensive and time consuming. Hence, it is infeasible to conduct subjective tests that can provide complete $q_{mos}$ model coverage for all the possible values and combinations of network factors in their GAP performance levels. In our closed-network testing, the $q_{mos}$ rankings were obtained for all the possible network condition combinations with one value of each network factor within each of the GAP performance levels. For this reason, we treat the $q_{mos}$ rankings from the closed-network testing as “training data”. On this training data, we use the statistical multiple regression technique to determine the appropriate closed-form expressions. Thus obtained closed-form expressions enable us to estimate the streaming and interactive GAP-Model $q_{mos}$ rankings for any given values and combinations of network factors measured in an online manner on a network path.

The average $q_{mos}$ ranking for a network condition $j$ ($q_{mos}^j$) is obtained by averaging the $q_{mos}$ rankings of the $N = 21$ human subjects for a network condition $j$ as follows:

$$q_{mos}^j = \frac{1}{N} \sum_{i=1}^{N} q_{mos}^{ij}$$

(4.1)

The $q_{mos}^j$ ranking is separately calculated for the streaming video clip tests (S-MOS) and the interactive video clip tests (I-MOS). This allows us to quantify the interaction difficulties faced by the human subjects in addition to their QoE when passively viewing impaired audio and video streams. Figure 4.6 illustrates the differences in the S-MOS and I-MOS rankings due to the impact of network factors. Specifically, it shows the decreasing trends of the S-MOS and I-MOS rankings for test cases with increasing values of $b_{net}$.
Figure 4.6: Comparison of Streaming MOS (S-MOS) and Interactive MOS (I-MOS)

$d_{\text{net}}$, $l_{\text{net}}$ and $j_{\text{net}}$ network factors. We can observe that at less severe network conditions ($< \text{GGGG} >$, $< \text{GAGG} >$, $< \text{GGAG} >$, $< \text{GAGA} >$, $< \text{GGPG} >$, $< \text{GGGP} >$), the decrease in the S-MOS and I-MOS rankings is comparable. This suggests that the human subjects’ QoE was similar with or without interaction in test cases with these less severe network conditions. However, at relatively more severe network conditions ($< \text{GAPG} >$, $< \text{GGPA} >$, $< \text{GGAP} >$, $< \text{GAPA} >$, $< \text{GGPP} >$, $< \text{AGPP} >$, $< \text{PPGA} >$, $< \text{PGPP} >$), the I-MOS rankings decrease quicker than the S-MOS rankings. Hence, the I-MOS rankings capture the perceivable interaction difficulties faced by the human subjects during the interactive test cases due to both excessive delays as well as due to impaired audio and video.

As explained in Section 4.2.2, the end-user QoE varies based on the users’ experience-levels with the VVoIP technology. Figure 4.7 quantitatively shows the differences in the average values of S-MOS rankings provided by the Expert, General and Novice human
Figure 4.7: Comparison of average S-MOS of Expert, General and Novice human subjects for Good $j_{net}$ performance level and with increasing network condition severity. Although there are minor differences in the average values for a particular network condition, we can observe that the S-MOS rankings generally decrease with the increase in network condition severity regardless of the human subject category.

To estimate the possible variation range around the average $q_{mos}$ ranking influenced by the human subject category for a given network condition, we determine additional $q_{mos}$ types that correspond to the 25th percentile and 75th percentile of the S-MOS and I-MOS rankings. We refer to these additional $q_{mos}$ types as “lower bound” and “upper bound” S-MOS and I-MOS. Figure 4.8 quantitatively shows the differences in the upper bound, lower bound and average values of S-MOS rankings provided by the human subjects for Good $j_{net}$ performance level and with increasing network condition severity. We observed similar differences in the upper bound, lower bound and average $q_{mos}$ rankings for both S-MOS and I-MOS under other network conditions as well.
Based on the above description of average, upper bound and lower bound $q_{mos}$ types for S-MOS and I-MOS rankings, we require six sets of regression surface model parameters for online estimation of GAP-Model $q_{mos}$ rankings. To estimate the regression surface model parameters, we observe the diagnostic statistics pertaining to the model fit adequacy obtained by first-order and second-order multiple-regression on the streaming and interactive $q_{mos}$ rankings in the training data. The diagnostic statistics in all cases show relatively higher residual error due to lack-of-fit and lower coefficient of determination (R-sq) values for the first-order multiple-regression compared to the second-order multiple-regression. Note that the R-sq parameter indicates how much variation of the response i.e., $q_{mos}$ is explained by the model. The R-sq values were less than 88% and greater than 97% in the first-order and second-order multiple-regression observations, respectively. Hence, the diagnostic statistics suggest that a quadratic model better represents the curvature in the I-MOS and S-MOS response surfaces than a linear model. Tables 4.2 and 4.3 shows the...
significant (non-zero) quadratic regression model parameters for the six GAP-Model $q_{mos}$ types, whose general representation is given as follows:

$$q_{mos} = C_0 + C_1 b_{net} + C_2 d_{net} + C_3 l_{net} + C_4 j_{net} + C_5 l_{net}^2 + C_6 j_{net}^2 + C_7 d_{net} l_{net} + C_8 d_{net} j_{net}$$

(4.2)

<table>
<thead>
<tr>
<th>Type</th>
<th>$C_0$</th>
<th>$C_1$</th>
<th>$C_2$</th>
<th>$C_3$</th>
<th>$C_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-MOS</td>
<td>2.7048</td>
<td>0.0029</td>
<td>-0.0024</td>
<td>-1.4947</td>
<td>-0.0150</td>
</tr>
<tr>
<td>S-MOS-LB</td>
<td>2.9811</td>
<td>0.0023</td>
<td>-0.0034</td>
<td>-1.8043</td>
<td>-0.0111</td>
</tr>
<tr>
<td>S-MOS-UB</td>
<td>1.7207</td>
<td>0.0040</td>
<td>-0.0031</td>
<td>-1.4540</td>
<td>-0.0073</td>
</tr>
<tr>
<td>I-MOS</td>
<td>3.2247</td>
<td>0.0024</td>
<td>-0.0032</td>
<td>-1.3420</td>
<td>-0.0156</td>
</tr>
<tr>
<td>I-MOS-LB</td>
<td>3.3839</td>
<td>0.0017</td>
<td>-0.0032</td>
<td>-1.3893</td>
<td>-0.0177</td>
</tr>
<tr>
<td>I-MOS-UB</td>
<td>3.5221</td>
<td>0.0021</td>
<td>-0.0026</td>
<td>-1.3050</td>
<td>-0.0138</td>
</tr>
</tbody>
</table>

Table 4.2: Regression surface model parameters $C_0$ to $C_4$ for the six GAP-Model $q_{mos}$ types

<table>
<thead>
<tr>
<th>Type</th>
<th>$C_5$</th>
<th>$C_6$</th>
<th>$C_7$</th>
<th>$C_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-MOS</td>
<td>0.2918</td>
<td>0.0001</td>
<td>0.0004</td>
<td>0.0055</td>
</tr>
<tr>
<td>S-MOS-LB</td>
<td>0.3746</td>
<td>0.0001</td>
<td>0.0005</td>
<td>0.0069</td>
</tr>
<tr>
<td>S-MOS-UB</td>
<td>0.2746</td>
<td>0.0001</td>
<td>0.0004</td>
<td>0.0043</td>
</tr>
<tr>
<td>I-MOS</td>
<td>0.2461</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0058</td>
</tr>
<tr>
<td>I-MOS-LB</td>
<td>0.2677</td>
<td>0.0001</td>
<td>0.0002</td>
<td>0.0055</td>
</tr>
<tr>
<td>I-MOS-UB</td>
<td>0.2614</td>
<td>0.0001</td>
<td>0.0001</td>
<td>0.0053</td>
</tr>
</tbody>
</table>

Table 4.3: Regression surface model parameters $C_5$ to $C_8$ for the six GAP-Model $q_{mos}$ types
4.1.4 GAP-Model Characteristics

In this section, we study the characteristics of the GAP-Model \( q_{mos} \) rankings and compare it with the \( q_{mos} \) rankings in the training data under the same range of network conditions.

Given that we have four factors \( b_{net}, d_{net}, l_{net} \) and \( j_{net} \) that affect the \( q_{mos} \) rankings, it is impossible to visualize the impact of all the factors on the \( q_{mos} \) rankings. For this reason, we visualize the impact of the training S-MOS and I-MOS rankings using an example set of 3-d graphs. The example set of graphs are shown in Figures 4.9 and 4.11 for increasing \( l_{net} \) and \( j_{net} \) values. In these Figures, the \( b_{net} \) and \( d_{net} \) values are in the Good performance levels and hence their effects on the \( q_{mos} \) rankings can be ignored. We can observe that each of the S-MOS and I-MOS response surfaces are comprised of only nine data points, which correspond to the three \( q_{mos} \) response values for GAP performance level values of \( l_{net} \) and \( j_{net} \) configured in the test cases. Expectedly, the \( q_{mos} \) values decrease as the \( l_{net} \) and \( j_{net} \) values increase. The rate (shown by the curvature) and magnitude (z-axis values) of decrease of \( q_{mos} \) values with increase in the \( l_{net} \) and \( j_{net} \) values is comparable in both the streaming and interactive test cases. Figures 4.10 and 4.12 show the corresponding GAP-Model S-MOS and I-MOS rankings for increasing values of \( l_{net} \) and \( j_{net} \) in the same ranges set in the training test cases. We can compare and conclude that the GAP-Model \( q_{mos} \) obtained using the quadratic fit follow the trend and curvilinear nature of the training \( q_{mos} \) noticeably.

To visualize the impact of the other network factors on the GAP-Model \( q_{mos} \), let us look at another example set of 3-d graphs shown in Figures 4.13 and 4.14. Specifically, they show the impact of \( l_{net} \) and \( d_{net} \) as well as \( l_{net} \) and \( b_{net} \) on the S-MOS rankings, respectively. Note that the 3-d axes in these graphs are rotated to obtain a clear view of
the response surfaces. We can observe from Figure 4.13 that the rate and magnitude of decrease in the $q_{mos}$ rankings is higher with the increase in $l_{net}$ values as opposed to the decrease with the increase in the $d_{net}$ values. In comparison, the rate of decrease in the $q_{mos}$ rankings with the decrease in the $b_{net}$ values as shown in Figure 4.14 is lower than with the increase in the $l_{net}$ values.
We remark that the above observations relating to the impact of network factors on the $q_{mos}$ rankings presented in this section are similar to the observations presented in related works such as [86] [3].

### 4.1.5 GAP-Model implementation in Vperf Tool

The GAP-Model framework implementation in the Vperf tool and the workflow to estimate VVoIP QoE is shown in Figure 4.15. We can see that the GAP-Model produces online VVoIP QoE estimates given: (i) the VVoIP session information and (ii) online network conditions. The VVoIP session information request (t) specifies the test session’s peak video encoding rate i.e., 256, 384 or 768 Kbps dialing speed, and whether the session involves streaming or interactive VVoIP streams. The online network conditions are measured by the VVoIP-session-traffic emulation tool we have developed called “Vperf” [96]. The Vperf tool generates low and high video $a_{lev}$ packet trains with packet sizes and inter-packet times corresponding to the VVoIP traffic model for a given request (t) session characteristics. If
the test request is of streaming type, Vperf generates probing packet trains only one-way (e.g., Side-A to Side-B in Figure 2), whereas, if the test request is of interactive type, Vperf generates probing packet trains both-ways i.e., between Side-A to Side-B and between Side-B to Side-A.

![Diagram of Online VVoIP QoE Measurement Framework](image)

Figure 4.15: Online VVoIP QoE Measurement Framework

After test duration $\delta t$ required to obtain a statistically stable measurement, the network condition measured in terms of network factors viz., bandwidth ($t+\delta t$), delay ($t+\delta t$), jitter ($t+\delta t$) and loss ($t+\delta t$) are input to the “GAP-Model”. The GAP-Model then produces a test report instantly with a VVoIP QoE estimate $MOS (t+\delta t)$ in terms of the GAP grades. Note that if the test request is of streaming type, the test report is generated on the test initiation side with the S-MOS rankings. However, if the test request is of interactive type, two test reports are generated, one at Side-A and another at Side-B with the corresponding side’s I-MOS rankings. The Side-A test report uses the network factor measurements collected on the Side-A to Side-B network path, whereas, the Side-B test report uses the network factor measurements collected on the Side-B to Side-A network path.
4.1.6 GAP-Model Validation

In this sub-section, we validate the GAP-Model $q_{mos}$ rankings using a new set of tests involving human subjects. In the new tests, we use network condition configurations that were not used for obtaining the training $q_{mos}$ rankings and thus evaluate the QoE estimation accuracy of the GAP-Model for other network conditions. We also compare the online GAP-Model $q_{mos}$ rankings with the $q_{mos}$ rankings obtained offline using the PSNR-mapped-to-MOS technique used in [78] and [79].

GAP-Model $q_{mos}$ Validation with Additional Human Subject Tests

As shown in Section 4.2.4, the GAP-Model $q_{mos}$ rankings are obtained by extrapolating the corresponding training $q_{mos}$ rankings response surfaces. Given that the training $q_{mos}$ rankings are obtained from human subjects for a limited set of network conditions, it is necessary to validate the performance of the GAP-Model $q_{mos}$ rankings for other network conditions that were not used in the closed-network test cases. For the validation, we conduct a new set of tests on the same network testbed and using the same measurement methodology described in Section 4.2.2. However, we make modifications in the human subject selection and in the network condition configurations.

For the new tests, we randomly select 7 human subjects from the earlier set of 21 human subjects. Recall, ITU-T suggests a minimum of 4 human subjects as compulsory for statistical soundness in determining $q_{mos}$ rankings for a test case. Also, we configure NISTnet with the randomly chosen values of network factors within the GAP performance levels as shown in Table 4.4. Note that these network conditions are different from the network conditions used to obtain the training $q_{mos}$ rankings. We refer to the $q_{mos}$ rankings
obtained for the new tests involving the Streaming-Kelly video sequence as “Validation-S-MOS” (V-S-MOS). Further, we refer to the $q_{mos}$ rankings obtained for the new tests involving the Interactive-Kelly video sequence as “Validation-I-MOS” (V-I-MOS).

<table>
<thead>
<tr>
<th>Network Factor</th>
<th>Good Level</th>
<th>Acceptable Level</th>
<th>Poor Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{net}$</td>
<td>100 ms</td>
<td>200 ms</td>
<td>600 ms</td>
</tr>
<tr>
<td>$l_{net}$</td>
<td>0.3 %</td>
<td>1.2 %</td>
<td>1.65 %</td>
</tr>
<tr>
<td>$f_{net}$</td>
<td>15 ms</td>
<td>40 ms</td>
<td>60 ms</td>
</tr>
</tbody>
</table>

Table 4.4: Values of network factors for GAP-Model validation experiments

Figures 4.16 and 4.17 show the average of the 7 human-subjects’ V-S-MOS and V-I-MOS rankings obtained from the new tests for each network condition in Table 4. We can observe that the V-S-MOS and V-I-MOS rankings lie within the upper and lower bounds and are close to the average GAP-Model $q_{mos}$ rankings for the different network conditions. Thus, we validate the GAP-Model $q_{mos}$ rankings and show that they closely match the end-user VVoIP QoE for other network conditions that were not used in the closed-network test cases.

**GAP-Model $q_{mos}$ Comparison with PSNR-mapped-to-MOS $q_{mos}$**

Herein, we compare the GAP-Model $q_{mos}$ rankings with the offline-type PSNR-mapped-to-MOS $q_{mos}$ (P-MOS) rankings used in [78] and [79]. The P-MOS rankings used are generated using the popular NTIA’s VQM software [32] that implements the algorithm ratified by ITU-T in their J.144 Recommendation [79] and the ANSI in their T1.801.03 Standard [84]. The VQM P-MOS rankings only measure the degradation of video pixels.
Figure 4.16: Comparison of S-MOS with Validation-S-MOS (V-S-MOS)

Figure 4.17: Comparison of I-MOS with Validation-I-MOS (V-I-MOS)
caused due to frame freezing, jerky motion, blurriness and tiling in the reconstructed video sequence and cannot measure interaction degradation. Hence, we only compare the GAP-Model S-MOS rankings with the VQM P-MOS rankings for different network conditions. To obtain the P-MOS rankings, we use the same network testbed that was used for the closed-network test cases and configure it with the network conditions shown in Table 4.4. For each network condition, we obtain 7 reconstructed Streaming-Kelly video sequences.

The process involved in obtaining the reconstructed Streaming-Kelly video sequences includes capturing raw video at the receiver-side using a video-capture device, and editing the raw video for time and spatial alignment with the original Streaming-Kelly video sequence. The edited raw video sequences further need to be converted into one of the VQM software supported formats: RGB24, YUV12 and YUY2. We choose the RGB24 video format. When provided with an appropriately edited original and reconstructed video sequence pair, the VQM software performs a computationally intensive per-pixel processing of the video sequences and produces a P-MOS ranking. Note that the above process to obtain a single reconstructed video sequence and the subsequent P-MOS ranking using the VQM software consumes several tens of minutes and requires a PC with at least 2 GHz processor, 1.5 GB of RAM and 4 GB free disk space.

Figure 20 shows the average of 7 P-MOS rankings obtained from VQM software processing of 7 pairs of original and reconstructed video sequences for each network condition in Table 4.4. We can observe that the P-MOS rankings lie within the upper and lower bounds and are close to the average GAP-Model S-MOS rankings for the different network conditions. Thus, we show that the online GAP-Model S-MOS rankings that are obtained almost instantly with minimum computation closely match the offline P-MOS rankings, which are obtained after a time-consuming and computationally intensive process.
4.2 VVoIP Interaction QoE Estimation

Videoconference sessions frequently experience performance failures due to changes in network conditions on the Internet. The session performance failures manifest to end-users as perceptual quality impairments such as video frame freezing and voice dropouts. These impairments cause extra end-user interaction effort and correspondingly lead to unwanted network bandwidth consumption that affects the overall Internet traffic congestion levels. Hence, the requirement in designing an efficient Internet videoconferencing system with optimum interaction QoE is to measure and subsequently minimize the extra end-user interaction effort in videoconference sessions. This observation along with an explanation on the need for schemes to measure human interaction difficulties in voice and video conferences is presented in [103]. In this section, we describe our novel “Multi-Activity
Packet-Trains” (MAPTs) methodology that considers end-user interaction effort and the corresponding network bandwidth consumption to measure objective VVoIP Interaction QoE. We also describe the MAPTs implementation in our Vperf tool and show Vperf’s measurements in a network testbed that features a wide variety of network fault events.

4.2.1 Low and High Video Activity Levels

Recall from Chapter 2 that video sequences can be broadly categorized as having either low or high $a_{lev}$. Low $a_{lev}$ video sequences feature slow body movements and a constant background (e.g. Claire video sequence). High $a_{lev}$ video sequences feature rapid body movements and/or quick scene changes (e.g. Foreman video sequence). In our study, we consider the ‘listening’ end-user action to produce a low $a_{lev}$ video sequence and the ‘talking’ end-user action to produce a high $a_{lev}$ video sequence.

![Figure 4.19: $b_{video}$ for Claire (low $a_{lev}$) video sequence](image)

Figure 4.19: $b_{video}$ for Claire (low $a_{lev}$) video sequence

To distinguish the low and high video $a_{lev}$ characteristics in terms of the bandwidth consumption $b_{video}$ in a videoconference session, let us look at Figures 4.19 and 4.20.
They show the instantaneous $b_{video}$ values for the Claire and Foreman video sequences, respectively for the common dialing speeds of the H.263 video codec. In the case of the Claire (low $a_{lev}$) video sequence, the full spread of video activity in the video sequence is realized in the encoding rates at the 256, 384 and 768 Kbps dialing speeds. Note that if we choose dialing speeds higher than 256 Kbps, the encoder attempts to utilize the extra bandwidth by encoding at marginally higher rates. Yet, the video activity is such that the encoder does not require all of the allowed bandwidth to produce a full spread of the video activity. In the case of the Foreman (high $a_{lev}$) video sequence, the video activity appears “clipped” for the 256 and 384 Kbps dialing speeds. Further, for the 768 Kbps dialing speed, the video activity appears clipped for the initial 4 seconds, after which, the full spread of the video activity is seen. This suggests that the encoder requires more than 768 Kbps bandwidth to produce a full spread of the video activity in the Foreman video sequence. If a lesser dialing speed is chosen by end-user, the encoder performs rate-limiting to limit the bandwidth consumption. In a nutshell, we can see that the bandwidth consumption for low
\( a_{lev} \) video i.e., ‘listening’ end-user action is less than that for high \( a_{lev} \) video i.e., ‘talking’ end-user action, irrespective of the session’s dialing speed. We will use this video encoding behavior observation in our MAFTs methodology explained in Section 4.3.3.

### 4.2.2 Network Fault Events

We now explain the characteristics of network fault events that affect the video \( a_{lev} \) during a videoconference session. The changes in the video \( a_{lev} \) in turn affect the \( b_{snd} \) traffic and cause unwanted interaction patterns between participants. The network fault events can occur in several different forms and timescales on the Internet. In measurement studies such as [22] and [23], the network fault events are found to occur in the form of bursts, spikes and other complex patterns and last anywhere between a few seconds to a few minutes. Considering the broad severity levels of the faults and the timescale within which end-point error-concealment schemes cannot ameliorate the voice and video degradation, we classify network fault events into two types: (i) Type-I, and (ii) Type-II. If the performance level of any network factor in the Good grade changes to the Acceptable grade over a 5 second period, we treat such an occurrence as a “Type-I” network fault event. Also, if the performance level of any network factor in the Good grade changes to the Poor grade over a 10 second period, we treat such an occurrence as a “Type-II” network fault event. Further, we consider the impact of a Type-I network fault event on end-user QoE to result in a ‘repeat’ action of a listening participant in a videoconference session. Along the same lines, we consider the impact of a Type-II network fault event on end-user QoE to result in the ‘disconnect’, ‘reconnect’ and ‘reorient’ actions between the participants in a videoconference session.
These ‘repeat’, ‘disconnect’, ‘reconnect’, and ‘reorient’ actions during a videoconference session are unwanted interaction patterns. This is because they increase the end-user effort required to complete a set of agenda-items in a videoconference session. In addition, video traffic in a videoconference session has high data rates (256 Kbps - 768 Kbps). Hence, if these unwanted interaction patterns occur at a large-scale in a videoconferencing system, they cause significant network bandwidth consumption. Such a consumption adds to the Internet traffic congestion levels and also adversely affect other application traffic. The end-user interaction effort and the corresponding network bandwidth consumption together can be measured using the “agenda-bandwidth” metric that we define as the aggregate network bandwidth consumed on both sides while completing a set of agenda-items in a videoconference session. Extra end-user interaction effort and corresponding network bandwidth consumption due to unwanted interaction patterns result in unwanted agenda-bandwidth. The ability to measure the unwanted agenda-bandwidth caused by the network fault events is vital in the frameworks. The measurements can be used to determine suitable traffic engineering techniques [104] that can foster efficient design of large-scale videoconferencing systems.

4.2.3 Multi-Activity Packet Trains (MAPTs)

To measure the unwanted agenda-bandwidth, it is not practical to use actual videoconferencing end-points and video sequences. This is because:

(a) end-point vendors do not provide APIs to collect real-time network performance measurements for network fault detection during a session due to proprietary reasons,

(b) it is non-trivial to dynamically modify the video sequences in end-points to emulate participant interaction patterns upon network fault detection, and
it is prohibitively expensive to deploy several end-points to simultaneously monitor multiple network paths.

Due to these and other repeatability reasons, it is preferable to deploy measurement servers that run active measurement tools which use schemes that can closely emulate the video-conferencing traffic characteristics to monitor videoconferencing interaction QoE related issues on the Internet.

Our “Multi-Activity Packet-Trains” (MAPTs) methodology is one such scheme that can be used to measure VVoIP interaction QoE affected by the Type-I and Type-II network fault events. The MAPTs methodology uses the active measurements principle where probing packet trains dynamically emulate participants’ interaction patterns and corresponding video activity levels in a videoconference session. To illustrate the interaction patterns with a simple example, an end-user’s ‘listen’ action followed by a ‘talk’ action would result in video activity levels emulated using back-to-back LH packet trains shown in Figure 4.21. Note that each packet train consists of a series of packets with $tps_i$ corresponding to the $i^{th}$ total packet size and $ipt_i$ corresponds to the $i^{th}$ inter-packet time.

![Figure 4.21: LH packet trains to emulate video activity levels for ‘listen’ followed by ‘talk’ end-user actions](image)

Note that each packet train consists of a series of packets with $tps_i$ corresponding to the $i^{th}$ total packet size and $ipt_i$ corresponds to the $i^{th}$ inter-packet time.
It should be noted that the below presented participant interaction patterns affected by network fault events are only a subset of the various possible patterns. Also, they have not been derived using actual human subject experiments and have been assumed based on typical considerations in a VVoIP system.

**Emulation of Participant Interaction Patterns**

We now describe the strategy to emulate participant interaction patterns (PIPs) used in MAPTs methodology. As shown in Figure 4.22, a general videoconference session between two or more participants will essentially have an “agenda” with several “items” to be covered over the session duration \( T \). The items correspond to different sets of activities to be accomplished by the participants with item completion times \( T_1, T_2, \ldots, T_n \). The beginning agenda-item is usually an introduction of the participants and in the concluding agenda-item, the participants usually summarize the discussions in the previous agenda-items and disperse.

![Figure 4.22: General videoconference session agenda-items](image)

For the sake of simplicity in explaining the MAPTs methodology, we assume that a videoconference session involves a participant on side-A asking a series of questions to another participant on side-B. Each question or answer corresponds to a separate agenda-item.
Further, if the participant on side-A is ‘talking’, we assume the participant on side-B to be ‘listening’ and vice versa. Hence, the total network bandwidth consumed by the participant on side-A asking questions, can be considered as the *request* of the videoconference session. Likewise, the total network bandwidth consumed by the participant on side-B while responding to the questions (or while satisfying the request) can be considered as the *response* of the videoconference session. If network fault events occur frequently in the session, side-A participant’s effort level increases and so does the request level. High request levels in videoconference sessions in a reasonably large-scale videoconferencing system obviously leads to high response levels, which can significantly contribute to the Internet traffic congestion levels and thus adversely affect other application traffic.

For such a videoconference session, we consider three different PIPs: $PIP_1$, $PIP_2$, and $PIP_3$. The $PIP_1$ corresponds to the participant interaction pattern when no network fault events occur during the videoconference session. Figure 7 shows the request and response for the $PIP_1$ interaction pattern. Recall from Section 4.3.1, a ‘listening’ participant’s video sequence produces low $a_{lev}$ characteristics, whereas, a ‘talking’ participant’s video sequence produces high $a_{lev}$ characteristics. The videoconference session starts with the participant on side-A doing the ‘talking’ for the introduction agenda-item. Following this, the agenda-items progress with each side participant ‘talking’ alternately without any network fault event interruptions. Finally, the videoconference session ends with the both-side participants ‘talking’ during the conclusion agenda-item.

The $PIP_2$ corresponds to the participant interaction pattern when a Type-I network fault event occurs during the videoconference session. Recall from Section 4.3.2 that such an event occurrence causes a ‘repeat’ action between the participants. Figure 4.24 shows the effects of the Type-I network fault event (occurring during agenda-item 2) on the request
and response in the videoconference session. We can see that once the Type-I network fault event affects the ‘listening’ side-A participant at time $T'_{\text{event}}$, the participant begins ‘talking’ to interrupt the ‘talking’ side-B participant and requests for a repeat of the previous statements. The time between $T'_{\text{event}}$ and $T_{\text{repeat}}$ corresponds to the time taken for the participant on side-B to complete responding to the repeat request made by the side-A participant. The revised time to finish the item 2 in this case is $T'_2$.

The $PIP_3$ corresponds to the participant interaction pattern when a Type-II network fault event occurs during the videoconference session. Recall from Section 2 that such an event occurrence causes ‘disconnect’, ‘reconnect’ and ‘reorient’ actions between the participants. Figure 4.25 shows the effects of the Type-II network fault event (occurring during agenda-item 2) on the request and response in the videoconference session. We can see that once the Type-II network fault event affects the ‘listening’ side-A participant at time $T''_{\text{event}}$, the participant begins ‘talking’ to interrupt the ‘talking’ side-B participant and requests for a session reconnection. The times $T_{\text{disconnect}}$, $T_{\text{reconnect}}$ and $T_{\text{reorient}}$, correspond to the
Figure 4.24: Videoconference session demand and supply for $PIP_2$

‘disconnect’, ‘reconnect’ and ‘reorient’ actions, respectively. The revised time to finish the item 2 in this case is $T''_2$.

Figure 4.25: Videoconference session demand and supply for $PIP_3$
Now that we have seen how to measure the request and response for a videoconference session affected by network fault events, our next step is to determine the “unwanted” request and response (marked in Figures 4.24 and 4.25). This will allow us to obtain the measure of unwanted agenda-bandwidth in the session. Since the agenda-bandwidth is the aggregate network bandwidth consumed on both sides in a videoconference session, it is calculated as a sum of all the instantaneous request and response bandwidth values over the session duration. The agenda-bandwidth measurements for $\text{PIP}_1$, $\text{PIP}_2$, and $\text{PIP}_3$ are shown in Figure 4.26. In the case of $\text{PIP}_1$, the agenda-bandwidth increases steadily and all the agenda-items consume $B_n$ bandwidth over agenda-time $T_n$. We treat this agenda-bandwidth $B_n$ over agenda-time $T_n$ as our baseline. In the cases of $\text{PIP}_2$ and $\text{PIP}_3$, the unwanted request and response increase the agenda-bandwidth to $B'_n$ over agenda-time $T'_n$ and $B''_n$ over agenda-time $T''_n$, respectively. The goal in designing an efficient Internet videoconferencing system will be to bring the $B'_n$ and $B''_n$ values as close as possible to the baseline $B_n$ using suitable traffic engineering techniques [104]. Thus, we can minimize the unwanted agenda-bandwidth and hence improve end-user interaction QoE in the videoconferencing system.

**Emulation of Video Activity Levels**

We now describe the videoconferencing traffic model to be used by the probing packet trains that emulate the low and high $a_{lev}$ video at the different dialing speeds: 256, 384 and 768 Kbps. Although there have been earlier works such as [106] [107] and [108] that model videoconferencing traffic characteristics, they consider encoders such as NV, BVC and CELLB found in early videoconferencing end-points such as LBNL’s Vic [109]. Hence, their modeling results are not representative of the source-traffic characteristics in
today’s end-points that use the H.263 codec\textsuperscript{6}. Recently, studies such as [110] have modeled the H.263 codec source-traffic from newer commercial end-points. However, they do not separately model the low and high $a_{lev}$ video traffic characteristics.

Since earlier proposed videoconferencing traffic models cannot be applied for our emulation of probing packet trains that emulate the low and high $a_{lev}$ video, we derive a videoconferencing traffic model. For this, we use a trace-analysis based approach, where we model several combined traces of popular video sequences used in the multimedia research community. In the following, we describe the video sequence traces used for deriving our traffic model and the obtained traffic model parameters.

\textsuperscript{6}Majority of today’s videoconferencing end-points use the H.263 video codec and a small fraction of the latest end-points support the H.264 video codec, which is an enhanced version of the H.263 codec targeted mainly for improved codec performance at low bit rates.
Video Sequence Traffic Traces

There are several video sequences available at [64] that are repeatedly used for purposes that include comparing compression schemes of video codecs and studying the effect of frame loss during transport on the receiver-side video distortion [63] [111] [112]. Given that the statistical characteristics of the video content (e.g. mean and covariance of frame quality) of these sequences are widely known, researchers use these video sequences as a reference to compare the performance of their proposed techniques with other existing techniques. We use several of these video sequences to obtain traffic traces for constructing our low and high $a_{lev}$ video traffic models at the different dialing speeds. Specifically, we choose a set of 10 video sequences with 5 video sequences (Grandma, Kelly, Claire, Mother/Daughter, Salesman) belonging to the low $a_{lev}$ category and 5 video sequences (Foreman, Car Phone, Tempete, Mobile, Park Run) belonging to the high $a_{lev}$ category. The video sequences within a category are combined and used in a videoconference session initiated on an isolated LAN testbed with two Polycom View Station end-points that use the H.263 video codec. The $b_{video}$ values for low and high $a_{lev}$ traces for the common dialing speeds are shown in Figures 4.27 and 4.28, respectively.

To determine the traffic-model that enables emulation of the time-series characteristics of the $b_{video}$ values using probing packet trains, we need to know the instantaneous $tps$ and $b_{video}$ values. By dividing the instantaneous $tps$ with the corresponding instantaneous $b_{video}$ values, we can obtain the instantaneous $ipt$ values. Given the instantaneous $tps$ and $ipt$ values for low and high $a_{lev}$ video packet trains, we can generate the MAPTs as illustrated earlier in Figure 4.21.
Figure 4.27: $b_{\text{video}}$ for combined low $a_{\text{lev}}$ video sequences

Figure 4.28: $b_{\text{video}}$ for combined high $a_{\text{lev}}$ video sequences
4.2.4 Video Packet Size Distribution Analysis

We now perform a distribution-fit analysis on the packet sizes in the traces to derive the \textit{tps} characteristics for the low and high \textit{a} \textsubscript{lev} video packet trains. Our distribution-fit analysis shows that the video \textit{tps} distribution corresponds to the Gamma distribution. Figure 4.29 shows an example illustration of the Gamma distribution fit for the high \textit{a} \textsubscript{lev} video at 256 Kbps dialing speed. Table 4.5 shows the Gamma distribution parameters obtained using the statistical “goodness of fit” testing for the \textit{tps} data in the low and high \textit{a} \textsubscript{lev} video traffic at different dialing speeds. The \( \alpha \) and \( \beta \) parameters correspond to the shape and scale parameters, respectively of the Gamma distribution whose probability distribution function is given as follows:

\[
F(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} x^{\alpha - 1} e^{-\frac{x}{\beta}}
\]

(4.3)

Figure 4.29: \textit{tps} distribution for high \textit{a} \textsubscript{lev} at 256 Kbps dialing speed; \( \alpha = 4.321, \beta = 110 \)
<table>
<thead>
<tr>
<th>Activity Level</th>
<th>Dialing Speed (Kbps)</th>
<th>α</th>
<th>β</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>256</td>
<td>4.115</td>
<td>102.0</td>
</tr>
<tr>
<td>Low</td>
<td>384</td>
<td>2.388</td>
<td>250.3</td>
</tr>
<tr>
<td>Low</td>
<td>768</td>
<td>1.625</td>
<td>240.1</td>
</tr>
<tr>
<td>High</td>
<td>256</td>
<td>4.321</td>
<td>110.0</td>
</tr>
<tr>
<td>High</td>
<td>384</td>
<td>1.517</td>
<td>281.6</td>
</tr>
<tr>
<td>High</td>
<td>768</td>
<td>1.142</td>
<td>446.1</td>
</tr>
</tbody>
</table>

Table 4.5: Gamma distribution parameters for low and high $a_{lev}$ video packet sizes

4.2.5 Video Encoding-Rate Time Series Analysis

We now perform time series modeling to derive the trend parameters for the video encoding rates shown in Figures 4.27 and 4.28. For this, we use the classical decomposition model [113] given as follows:

$$X_t = m_t + Y_t$$  \hspace{1cm} (4.4)

where, $m_t$ is the trend component and $Y_t$ is the random noise component with $E[Y_t] = 0$. To obtain the trend $m_t$, we apply the differencing technique and separate out the residual noise component $Y_t$. Figures 4.30(a) and (b) show the autocorrelation functions (ACFs) of the $m_t$ and $Y_t$ components after applying lag-2 differencing. The ACF of $m_t$ has a sharp cut off at lag-2 and thus suggests a second-order moving average (MA(2)) process. Further, the ACF of $Y_t$ clearly indicates an i.i.d. noise process i.e., we have successfully captured the trend as an MA(2) process.

Table 4.6 shows the model parameters obtained by performing time series modeling of the $b_{video}$ for the low and high $a_{lev}$ video traffic at 256, 384 and 768 Kbps dialing speeds. The $\theta_1$, $\theta_2$, $\mu$ and $\sigma^2$ parameters correspond to the MA1 and MA2 co-efficients, mean and
<table>
<thead>
<tr>
<th>Activity Level</th>
<th>Dialing Speed (Kbps)</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\mu$</th>
<th>$\sigma^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>256</td>
<td>-1.2395</td>
<td>0.2395</td>
<td>192</td>
<td>200</td>
</tr>
<tr>
<td>Low</td>
<td>384</td>
<td>-1.1614</td>
<td>0.1614</td>
<td>253</td>
<td>400</td>
</tr>
<tr>
<td>Low</td>
<td>768</td>
<td>-1.2684</td>
<td>0.2684</td>
<td>301</td>
<td>500</td>
</tr>
<tr>
<td>High</td>
<td>256</td>
<td>-1.5213</td>
<td>0.5213</td>
<td>249</td>
<td>5</td>
</tr>
<tr>
<td>High</td>
<td>384</td>
<td>-1.4741</td>
<td>0.4741</td>
<td>349</td>
<td>140</td>
</tr>
<tr>
<td>High</td>
<td>768</td>
<td>-1.3024</td>
<td>0.3024</td>
<td>720</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 4.6: MA(2) parameters for low and high $a_{lev}$ video encoding rates

variance, respectively of the MA(2) process given as follows:

$$X_t = Z_t + \theta_1 Z_{t-1} + \theta_2 Z_{t-2}$$  \hspace{1cm} (4.5)

where $Z_t$ is an i.i.d. noise process with mean 0 and variance $\sigma^2$.

Figure 4.30: (a) ACF of $m_t$ (b) ACF of $Y_t$ - for high $a_{lev}$ at 256 Kbps dialing speed

To confirm this and to verify the order of the MA process, we apply the “Akaike Information Criterion” (AIC) test on the differenced trend data. Figure 4.31 shows the AIC
values for different MA order $q$ in [1...10]. We can observe a notable dip in the AIC value at $q = 2$ confirming that the process is indeed MA(2). Although, we notice lower AIC values for $q > 3$, the dip is not significant compared to the $q = 2$ case. This suggests that using a higher order model unnecessarily increases the complexity of the model.

Figure 4.31: AIC for high $a_{lev}$ at 256 Kbps dialing speed
4.2.6 MAPTs implementation in Vperf Tool

In this section, we describe the implementation of the Vperf tool that generates the MAPTs to dynamically emulate a videoconference session’s participant interaction patterns and corresponding video activity levels that are affected by network fault events. The tool components and their workflow are shown in Figure 4.32. The inputs to the Vperf tool include the session agenda, dialing speed i.e., 256, 384 or 768 Kbps and videoconference session type i.e., streaming or interactive. The session agenda consists of the $L$ and $H$ video $a_{lev}$ packet train order and their lengths corresponding to the agenda-items. For the streaming type, only one side’s packet train information is specified, whereas, both side’s packet train information needs to be specified for the interactive type.

A simple example session agenda for interactive type is shown in Figure 4.33. It consists of three sections that correspond to the three PIPs: $PIP_1$, $PIP_2$, and $PIP_3$. Each row in the $PIP_1$ section corresponds to a particular agenda-item. The $PIP_2$ section row is used to specify the length of the $H$ video $a_{lev}$ packet train to emulate a ‘repeat’ action if a Type-I network fault event is detected. Similarly, the $PIP_3$ section rows are used to specify the length and video $a_{lev}$ of the packet trains to emulate the ‘disconnect’, ‘reconnect’ and ‘re-orient’ actions if a Type-II network fault event is detected. Note that the $N$ during the reconnect action corresponds to no video $a_{lev}$ portion of the session.

If the session specification is done at Side-A and is of the streaming type, the Vperf generates $b_{video}$ only one-way from Side-A to Side-B according to the specified session agenda and dialing speed parameters. If the session specification is of the interactive type, the Vperf generates $b_{video}$ from Side-A to Side-B and vice versa according to the specified session agenda and dialing speed parameters. For a particular $L$ or $H$ video $a_{lev}$ specified in
Figure 4.32: Vperf tool components and their workflow

<table>
<thead>
<tr>
<th>PIP-Type</th>
<th>Side-A Train ($a_{lev}$)</th>
<th>Side-B Train ($a_{lev}$)</th>
<th>Train Duration (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PIP_1$</td>
<td>$L$</td>
<td>$H$</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>$H$</td>
<td>$L$</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$L$</td>
<td>$H$</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>$H$</td>
<td>$L$</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$L$</td>
<td>$H$</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>$H$</td>
<td>$L$</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$L$</td>
<td>$H$</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>$H$</td>
<td>$L$</td>
<td>20</td>
</tr>
</tbody>
</table>

$PIP_2$  

$PIP_3$  

Figure 4.33: An example Vperf tool session agenda specification

the session-agenda, the Vperf tool uses the tps and ipt traffic model parameters (obtained from Sections 4.3.3 and 4.3.4) specified in the “Traffic Model” file.
For these inputs, the outputs from Vperf tool are as follows: Based on the emulated traffic performance, Vperf continuously collects online measurements of $b_{net}$, $d_{net}$, $f_{net}$, and $l_{net}$ network factors on a per-second basis and appends it to an “Interim Test Report”. It also produces a total average of these measurements at the end of the session in the form of a “Final Test Report”. For an interactive session type, it also outputs the agenda-bandwidth and agenda-time measurements in the final test report. To generate these measurements for an emulated session, it uses the “Interaction Behavior Controller” component. This component processes the interim test report and detects the occurrence of Type-I and Type-II network fault events by looking up the detection rules (refer to Table 2.6.1) specified in the “Network Factor Limits” file. When a Type-I or Type-II network fault event occurrence is detected, the interaction behavior controller alters the emulation of the session agenda based on the participant interaction patterns explained in Section 4.3.3. In our current implementation, if 5 consecutive network fault events occur back-to-back, the session is terminated and the network path is deemed unusable for videoconferencing in the final test report. Thus, after completion of the session emulation as per the session-agenda, the VVoIP interaction QoE is reported by the Vperf tool based on the progress of the session-agenda.

### 4.2.7 MAPTs Evaluation

In this section, we first evaluate the MAPTs emulation characteristics of the Vperf tool. Figures 4.34, 4.35 and 4.36 show the characteristics of the $LH$ packet trains emulation by the Vperf tool at the 256, 384 and 768 Kbps dialing speeds, respectively. These characteristics have been collected at the sender-side of the Vperf tool in an isolated LAN environment. We can see that the mean and spread of the $LH$ packet trains are similar to the low
LH packet trains emulation by Vperf at 256 Kbps dialing speed and high video $a_{lev}$ model characteristics shown in Table 4.6. Further, we verified that the packet size distribution of the LH packet trains has the Gamma distribution parameters for low and high $a_{lev}$ video packet sizes shown in Table 4.5. Thus, we show that the Vperf tool emulation in practice is conformant with our videoconferencing traffic model parameters.

Now, we describe the videoconference session performance measurements collected on an isolated network testbed consisting of two measurement servers separated by the NISTnet network emulator [65]. The NISTnet was dynamically configured with different WAN profiles that featured the occurrence of varying number of Type-I and Type-II network fault events during videoconference session emulation by the Vperf tool. The session-agenda input to the Vperf tool contained the packet trains video $a_{lev}$ and train durations for the $PIP_1$, $PIP_2$ and $PIP_3$ cases as shown in Figure 4.33.

Figures 4.37 and 4.38 show the unwanted agenda-bandwidth measurements from the Vperf tool for increasing number of NISTnet generated Type-I and Type-II network fault events, respectively. Each measurement is an average of 10 emulation runs on the isolated network testbed. As expected, the amount of unwanted agenda-bandwidth is zero when
Figure 4.35: LH packet trains emulation by Vperf at 384 Kbps dialing speed

Figure 4.36: LH packet trains emulation by Vperf at 768 Kbps dialing speed

there is no network fault event occurrence and the unwanted agenda-bandwidth increases almost linearly with the number of network fault events occurrence. Given that the bandwidth consumed during ‘disconnect’, ‘reconnect’ and ‘reorient’ actions upon occurrence of a Type-II network fault event is higher than the bandwidth consumed in just a ‘repeat’ action upon occurrence of a Type-I network fault event, we can see that the unwanted
Figure 4.37: Impact of increasing number of Type-I network fault events on Agenda-bandwidth

Agenda-bandwidth is greater for Type-II network fault event cases than Type-I network fault event cases regardless of the dialing speed.

Further, the effects of the increase in the Type-I and Type-II network fault events on the unwanted agenda-time at the 768 Kbps dialing speed are shown in Figure 4.39. Similar unwanted agenda-time characteristics were noticed at the 256 Kbps and 384 Kbps dialing speeds also. Again, we can observe almost a linear increase in the unwanted agenda-time with the increase in the number of network fault events occurrence. Given that the time involved in the ‘disconnect’, ‘reconnect’ and ‘reorient’ actions upon occurrence of a Type-II network fault event is higher than the time involved in just a ‘repeat’ action upon occurrence of a Type-I network fault event, we can see that the unwanted agenda-time is greater for Type-II network fault event cases than Type-I network fault event cases regardless of the dialing speed.
Figure 4.38: Impact of increasing number of Type-II network fault events on Agenda-bandwidth

Figure 4.39: Impact of increasing number of Type-I and Type-II network fault events on Agenda-time at 768 Kbps dialing speed

4.3 Summary

In this chapter, we described our GAP-Model and MAPTs methodology implemented in the Vperf tool we developed for measuring streaming and interaction VVoIP QoE over
the Internet. We explained how these techniques were online in nature and did not require: (a) end-users to provide quality rankings, (b) actual video-sequences, and (c) expensive VVoIP end-points. We formulated the GAP-Model’s closed-form expressions based on an offline closed-network test methodology involving 21 human subjects. The human subjects ranked QoE of streaming and interactive video clips in a testbed featuring all possible combinations of the network factors in their GAP performance levels. The offline closed-network test methodology leveraged test case reduction strategies that significantly reduced a human subject’s test duration. Yet, it did not compromise the rankings data required for adequate model coverage. We validated the GAP-Model with additional human subject tests for network conditions that were not used in the closed-network testing. We also showed that the GAP-Model’s online estimates of VVoIP QoE are comparable to the J.144 estimates that are produced offline using a computationally intensive and time consuming process. In the last part of this chapter, we explained our novel MAPTs methodology to measure interaction VVoIP QoE. The MAPTs dynamically emulated participants’ interaction patterns and corresponding video activity levels in a videoconference session that are affected by network fault events. We also detailed the characteristics of the probing packet trains in MAPTs. For this, we used a videoconferencing traffic-model derived from the trace-analysis of video sequences widely used in the multimedia research community. We proposed two novel interaction QoE metrics viz., unwanted agenda-time and unwanted agenda-bandwidth, and evaluated their performance for increasing number of network fault events detected on a network path.
CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this chapter, we summarize our dissertation contributions, discuss open issues and suggest future directions. In Section 5.1, we describe the important problems we have solved and list the major dissertation contributions. While doing so, we explain how our contributions enable frameworks that manage networked computing resources to become ‘network QoS-aware’ and ‘end-user QoE-aware’. We also explain how our solutions were implemented and tested in both controlled LANs as well as on the Internet. In Section 5.2, we list the several open issues in our work. In the same discussion, we also present some future directions that can address the open issues to ensure the success of advanced Internet applications in the coming years.

5.1 Contributions

Active measurements are being widely collected on the Internet because they provide end-to-end network QoS information. The collected measurements are vital in network control and management frameworks that support advanced Internet applications. They can be used to monitor and predict network health, and to estimate end-user QoE for both pro-active and reactive purposes without much manual effort. The pro-active purposes include selecting network paths between networked computing resources that ensure satisfactory
end-user QoE. The reactive purposes include isolating and troubleshooting bottleneck hops that are impacting end-user QoE. In this dissertation, we have conducted a comprehensive study of various issues related to provisioning and analysis of active measurements in the frameworks. Our contributions listed below enable the frameworks to identify network bottlenecks and adapt application resources in a manner that guarantees end-user satisfaction when using advanced Internet applications.

**Improving Provisioning of Internet measurements**

Our first contribution is in identifying the measurement scheduling constraints that need to be addressed while satisfying on-going and on-demand measurement requirements of frameworks. A major constraint we discovered is the measurement conflict problem, which was mostly overlooked in previous works. We demonstrated the importance of conflict avoidance using several experiments involving popularly used measurement tools such as Iperf, H.323 Beacon, and Pathload on real networks. Another major constraint we studied was measurement regulation, which can be implemented using the Measurement Level Agreement specification we defined. This specification ensures active measurements do not overly consume network resources needed by actual application traffic.

With the knowledge of these constraints, we developed heuristic bin-packing and EDF-CE measurement scheduling algorithms that not only addressed the constraints, but also notably improved the schedulability of on-going measurements by up to 300% compared to existing schemes such as token passing, and resource-broker that use the round-robin principle. Such an improvement in the schedulability was achieved due to our proposed concurrent execution principle that allowed overlap of measurements if they did not conflict with one another. We developed a recursive pushing scheme in our measurement scheduling algorithms that minimized the response time of on-demand measurements by 50% without
disrupting the on-going measurements and without violating the measurement constraints. Such an improvement in the response time was achieved due to the moving of offline jobs without violating their deadlines, and ultimately deriving maximum early slack in the offline schedule. In addition, we extended our scheduling algorithms with the Raymond’s algorithm to function in a distributed manner without requiring a central orchestration mechanism. Using extensive simulations, we showed that supporting distributed schedule management with our scheduling algorithms added very minimal scheduling overhead. Specifically, the scheduling overhead was on the order of a couple of seconds, which is insignificant when compared to the measurement time scales that are on the order of several minutes between jobs. Further, we dealt with technical and policy issues involved in provisioning measurements across multiple ISP domains in either a centralized or distributed manner. In this context, we discussed how our scheduling algorithms can be integrated into measurement federations that are beginning to emerge in communities such as Internet2, DANTE and Global Grid Forum to facilitate multi-domain measurements.

We implemented our measurement scheduling algorithms in the ActiveMon framework for network monitoring and the widely used NWS framework for network weather forecasting. Using our ActiveMon implementation on an Internet2 testbed, we verified that our scheduling measurements are always conflict-free. Using another academic backbone network testbed, we also assessed the ability of our scheduling algorithms to sample network status in a manner that allows detecting notable network performance anomalies due to network events such as route changes and rate-limit misconfigurations on the Internet. From our NWS implementation experience, we discovered that no measurement scheduling algorithm that addresses the measurement constraints can cater to the strict measurement periodicity required for accurate network weather forecasting. We overcame this limitation
using a simple piece-wise linear interpolation technique that transformed the provisioned measurements in NWS. The transformation produced relatively more accurate network weather forecasts with the mean square error reduced by almost a half when compared to the accuracy without transformation.

**Improving Analysis of Internet measurements**

Provisioning network QoS metrics as desired by the frameworks can make the frameworks ‘network QoS-aware’. However, they are not sufficient to estimate the effect of the network status on end-user QoE, especially when they involve VVoIP applications such as videoconferencing and video-on-demand. Our next set of contributions is focused on analysis techniques that can be coupled with the provisioned measurements data to enable the frameworks to become ‘end-user QoE-aware’.

A major analysis challenge is to obtain online estimates of end-user QoE on several network paths simultaneously based on the provisioned network QoS measurements data. In addition, end-user QoE varies based on whether the VVoIP application involves streaming VVoIP (e.g. video-on-demand) or interactive VVoIP (e.g. videoconferencing). Further, each of the network factors have a range of values, and combinations of these ranges pertaining to the different network factors thus present a very large number of network QoS conditions to be covered in the testing. The fact that involving actual end-users to provide subjective quality assessments is a prohibitively expensive and time-consuming process, it is impractical to execute all the test cases for each human subject. We addressed the test case reduction challenge using a novel offline closed-network testing methodology and reduced the test cases to only 42 each for streaming and interactive VVoIP. The reduced test cases were completed by all of the human subjects within an hour and a half testing time.
However, while reducing the test cases, we did not compromise the collection of the rankings data needed for adequate model coverage. From the human subject testing, we developed a novel psycho-acoustic/visual cognitive model called GAP-Model that can provide objective QoE estimates in an online manner. We note that existing online QoE estimation techniques such as E-Model are limited to just VoIP applications, whereas the GAP-Model covers VVoIP, which includes VoIP as well. The GAP-Model maps the combined effects of QoS metrics such as bandwidth, delay, jitter and loss to Good, Acceptable and Poor (GAP) grades of end-user QoE for both streaming and interactive VVoIP. Using validation studies, we showed that MOS rankings obtained using the offline PSNR-based J.144 method were within the upper and lower bounds of the GAP-Model, which were output almost instantly. In comparison, the offline method required both original and reconstructed video sequences to be available for analysis and also involved extensive post-processing of the video sequences that required a computationally intensive and time consuming process.

We also developed an alternate approach to objectively measure interaction VVoIP QoE in terms of the interaction difficulties experienced by end-users on a network path. This approach involved Multi-Activity Packet Trains (MAPTs) that mimic a videoconference session’s participant interaction patterns and the corresponding video activity levels affected by intermittent network fault events. To facilitate the MAPTs emulation, we developed an empirical videoconferencing traffic model from the trace-analysis of video sequences widely used in the multimedia research community. We verified the MAPTs methodology in a network testbed that featured a wide variety of network fault events. We found that irrespective of the dialing speed, unwanted agenda-bandwidth and unwanted agenda-time increased significantly with the increase in the number of network fault event occurrences.
Both our GAP-Model and MAPTs have been implemented in a measurement tool we developed called Vperf that can be scheduled using our measurement scheduling algorithms for online VVoIP QoE estimation in NMIs.

5.2 Open Issues and Future Directions

The area of research that deals with network control and management involving active measurement principles is fairly new. Researchers in collaboration with ISPs have been collecting and analyzing active measurements using tools such as Ping and Traceroute for quite a while. However, only recently there has been sustained interest in developing measurement tools that use sophisticated probing techniques to measure the complex Internet dynamics. Also, with the recent emergence and rapid deployment of advanced Internet applications such as VoIP and videoconferencing, collecting network-wide network QoS measurements and mapping them to end-user QoE has become critically important. Therefore, this dissertation has addressed only a small subset of the numerous challenges involved in provisioning and analysis of Internet measurements for adaptation purposes in automated frameworks.

Semantic Priority Scheduling

One of the issues we did not consider in our measurement scheduling algorithms in Chapter 3 is the “semantic priority” of measurement requests. Tagging a semantic priority along with a measurement request allows distinguishing measurement tasks based on whether they correspond to intra-domain or inter-domain requests. The ability to distinguish between tasks can be particularly useful in deciding which kind of measurement requests can be blocked when measurement resources are not sufficient to meet all of the measurement requirements. The semantic priorities can also be used within a particular
domain to enforce MLAs that reserve a certain percentage of the measurement time for intra-domain requests. Further, they can be used to adjust the blocking probability of a particular ISP’s measurement requests amongst all of the inter-domain measurement requests.

**Policy-driven Scheduling**

Another issue that we did not address in-depth relates to the measurement policy specifications and their enforcement in multi-domain measurement scheduling. We addressed simple policy issues such as specifying MLAs in percentage or bits per second using our scripting language interface; the MLAs were later enforced by our scheduling algorithms for measurement regulation. However, in reality there are several other policy issues (e.g. encryption policy, role-based policy, time/date-based policy) that deal with different aspects of access control of the measurement resources and network resource consumption. Given that these policy issues are dynamic and their enforcement needs to include audit trails, adopting policy languages such as OASIS XML Access Control Markup Language (XACML) can be pertinent. In addition, coupling them with our scheduling algorithms with suitable heuristics can assure proper enforcement of the policies. With such extensions to our work, unaddressed hurdles involved in dealing with and enforcing multi-domain measurement policy issues can be overcome.

**Automated Anomaly Detection**

In our ActiveMon case study in chapter 3, we presented several kinds of network performance anomalies that we noticed on academic backbone networks. Most of these anomalies were detected by manually analyzing our measurement data sets. Since such a manual process is tedious and time-consuming, our future work is to investigate automated techniques for rapidly detecting anomalies and more importantly network fault events in our
measurement data. For this, we would like to employ and extend algorithms such as the “plateau” algorithm used in NLANR AMP and the “percentiles-threshold” algorithm used in RIPE NCC. Our ultimate goal is to quantify a particular network path’s reliability based on the severity and frequency of the network fault events. Treating a network path as a repairable system and using repair rate models is one of the ways we plan to use to quantify the reliability.

**GAP-Model Extensions**

The closed-network experiments to derive the GAP-Model in chapter 4 were primarily targeted towards the H.263 video and G.722 voice codecs. These codecs were chosen because they are most commonly used in videoconferencing end-points today. However, the same experiment methodology can be applied to derive additional variants of the GAP-Model closed-form expressions for low bit rate voice codecs such as GSM and loss-tolerant video codecs such as H.264 used in hand-holds. The end-user QoE will be different for these codecs because they handle the effects of network performance bottlenecks differently from the H.263 and G.722 codecs. Thus, we can model the differences in the QoE for different voice and video codecs under the same network conditions. Such a knowledge of the differences can be leveraged for “learn on-the-fly” adaptation of codec bit rates and codec selection in end-points and MCUs for delivering optimal end-user QoE.

**MAPTs Extensions**

The MAPTs methodology presented in Chapter 4 is a preliminary study. This is because - only basic participant interaction patterns and network fault event types were considered. In reality, there are several others that commonly occur in Internet videoconferencing. For instance, the participant interaction pattern in cases where there is a lack of
lip-synchronization can be considered. Also, the duration of the repeat/reorient actions when a network fault event involving excessive packet reordering can be included in the MAPTs methodology. Further, Human Visual System (HVS) principles can be adopted to classify the participant interaction patterns under different network fault event conditions. Coupling these studies with human subject experiments can more accurately map and validate the network fault event types and participant interaction patterns. The obtained results can then implemented into the Vperf tool for Internet measurements in NMIs.
BIBLIOGRAPHY


[61] “The Video Development Initiative (ViDe) Videoconferencing Cookbook” - http://www.vide.net/cookbook


[76] Telchemy VQMon - http://www.telchemy.com


169


