

# The management and processing of network performance information

O Bashir, I Phillips, D Parish, J L Adams and T Spencer

The performance of communications networks can be monitored intrusively by transmitting test packets on the network links being tested. The delays experienced by these test packets and the number of test packets lost or duplicated may provide an indication of the performance of the links being monitored. Intrusive network monitoring, even at modest sampling rates, generates a significantly large amount of primitive data. This data needs to be summarised appropriately to address the required network performance analysis operations. These operations are generally iterative in nature, where the results of initial processing and analysis generate requirements for further processing and analysis. This paper describes the architecture of a database that can assist in efficient analysis of network performance by reusing the information derived at various stages.

## 1. Introduction

Intrusive network performance monitoring involves making probing requests over the network elements being monitored. Time-stamped test packets (Fig 1) are transmitted over the links being monitored by a transmitting test station towards a receiving test station which logs their arrival, along with the time of their reception.

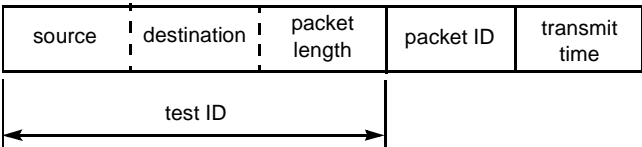


Fig 1 Test packet.

In order to perform accurate delay measurements, the clocks at the transmitting and the receiving test stations must be synchronised. This may be accomplished by acquiring timing information from external global timing agencies, for example the Rugby Clock [1, 2] and the global positioning system (GPS) [3, 4].

Data collection represents the lowest level of any monitoring activity [5]. The data collected by the test stations is regularly downloaded to a control and data-processing workstation where it is maintained in a suitable database. Analysts can then retrieve and process this historical data so that appropriate analysis activities may be performed. An architecture for such a system is shown in Fig 2. An elaborate network monitoring system may employ a large number of transmitting and receiving test stations,

which may test several routes of the network with a variety of transmission packet sizes.

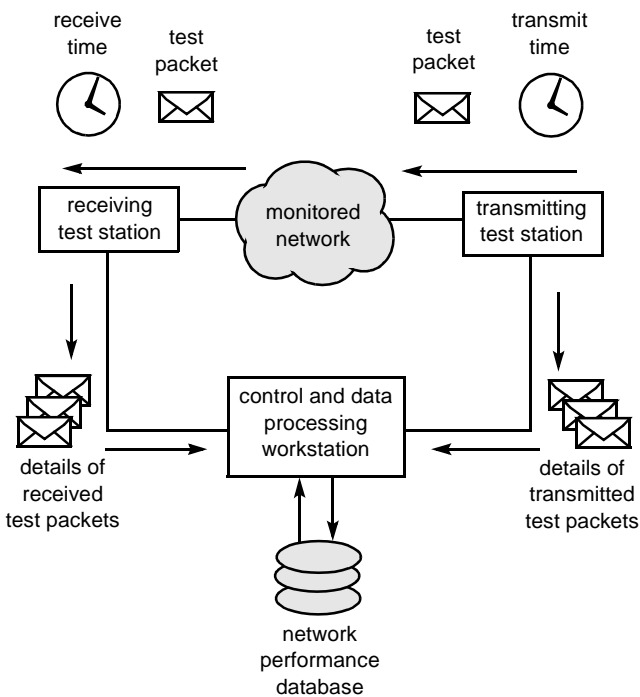


Fig 2 Network performance monitoring system.

Network performance analysis can be an iterative and an arduous activity. Typically, the analyst starts by processing the monitored data to generate a set of

summaries at a specific level of granularity. These summaries are then further analysed to detect various events, e.g. excessive delays, losses or duplication of test packets. In order to investigate these events in detail, the analyst may then need to vary the level of granularity as well as derive a different set of summaries. For example, it may be necessary to zoom into the event by generating summaries at a finer level of granularity. Alternatively the analyst may decide to zoom out and derive the same summaries at a coarser level of granularity (for a longer analysis period). This would allow the analyst to correlate the event with previous network effects. On the other hand, if the average delay for a certain analysis window is higher than a specific threshold, the analyst may want to know the average delay for, say, the slowest 10% test packets and the approximate range of delay values within that analysis window.

These iterative examples illustrate the need for a database system that can efficiently derive the required summaries. One obvious way of achieving this in an efficient manner is to reuse summaries derived at a finer level of granularity to derive the same summaries at the required level of granularity. This is an appropriate mechanism if the summaries are additive, or if the summaries can be decomposed into a set of common additive components. In this case these granular additive components may be reused to generate coarse granularity additive components which may then be processed to derive the required summaries [6–8].

Summaries related to the loss and duplication of test packets are additive in nature. The appropriate finer granularity loss and duplication counts can therefore simply be added to provide counts at the required level of granularity. Packet-delay summaries, however, are often based on the numerical order of the packet-delay values (e.g. percentiles) and are therefore neither additive nor can be decomposed into additive components. Such summaries usually need to be derived from the primitive data each time the analyst wants to vary the granularity.

This paper proposes a database architecture that structures and preprocesses the primitive network performance data so as to enhance its reusability. This is achieved by processing the primitive data into a generally useful intermediate form referred to here as intermediate information. Finer granularity intermediate information can then be combined to generate useful information at coarser levels of granularity. Intermediate delay information can then efficiently provide both non-additive as well as additive user-requested summaries. As access to the primitive data is reduced, the performance of the database is seen to increase significantly.

A prototype database has been developed in Java. It stores primitive data and the derived intermediate information as serialised objects [9]. An information system based on these concepts has been incorporated into a larger network monitoring system being developed at Loughborough University [10] as a part of BT's Managing Multiservice Networks (MMN) University Research Initiative project.

The next section of the paper describes the basic concepts of intermediate information. The structure of primitive performance data in the database is described in section 3. Section 4 discusses data structures suitable for network performance intermediate information. The generic architecture of the components that process and manage intermediate information is then described in section 5. The performance enhancements achieved as a result of reusing the network performance information are demonstrated in section 6. The paper is then concluded after describing a network performance information server.

## 2. Basic concepts of intermediate information

As has been suggested, the performance information requirements for different network management activities vary significantly. In certain cases (e.g. research and development) the processing requirements may be extremely dynamic and may not be known in advance. However, many of the applications processing network performance data proceed through a number of similar stages. Some applications may in fact require information that can be provided by other applications. In such situations it is tempting to access the information that has already been processed by these other applications to generate the required information (Fig 3).

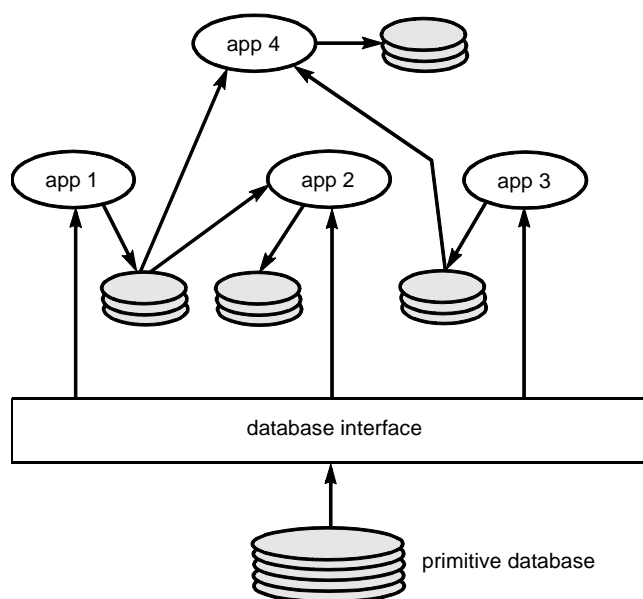


Fig 3 Information sharing across applications.

However, it is important to note that many information processing applications have a significantly shorter life than the data they process. Some of these applications are developed only to fulfil a one-off requirement. Others may be used rarely at specific instances. Some of the applications may evolve and provide information elements that are different from the ones provided by earlier versions. Therefore designing the information processing components of the monitoring system on the basis of the specific output of a few applications may be a short-sighted decision. If some applications change, others may be adversely affected.

Employing reusable information structures can, however, enhance the efficiency of the information processing algorithms and applications significantly. This is accomplished by preprocessing the primitive data so that it can fulfil multiple information requirements efficiently. These preprocessed information elements may be termed intermediate information.

These intermediate information elements are derived at the finest practical level of granularity and are cached in an intermediate information base (Fig 4). These elements can be combined appropriately to generate intermediate information at the courser levels of granularity.

As multiple information requirements can be fulfilled directly from the preprocessed intermediate information, access to the primitive database is reduced significantly, thus enhancing the efficiency of the information processing applications.

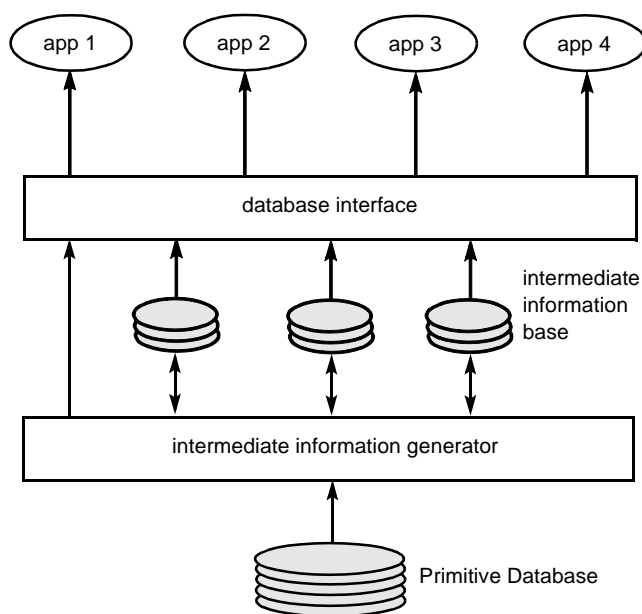


Fig 4 Information sharing across applications.

### 3. Structuring primitive network performance data

In its most basic form, primitive network performance data may be stored in the database as two separate tables — one table for the data representing the details of the transmitted test packets and the other for details of the received test packets. Processes attempting to derive packet delay summaries may need to access only the received data table. However, processes attempting to derive packet-count summaries need to access both tables and correlate the records to determine the conditions of packet loss and duplication. If a record exists in the transmitted data table representing a transmitted test packet for which there is no received data record, this indicates the loss of a test packet. Similarly, there may exist multiple test packets in the received data table with the same transmit time and packet identifier but different reception times. This indicates duplication of the test packet as it was communicated through the network (Fig 5).

Detection of packet loss and duplications from separate primitive database tables can therefore be a complex process. This is simplified by maintaining a single pre-joined table for both the transmitted and received test packets. Each record in this table contains the transmitted and received times for a test packet, its identifier and the delay experienced by the packet during communication (Fig 6). Computation of packet-delay values in advance can save a significant amount of later processing.

A primitive database manager object reads the log files retrieved from the test stations and creates pre-joined tables for each test. In the example system developed at Loughborough, the finest granularity for analysis reports is one hour, and a separate table is generated for every hour of the monitoring period. These tables are stored as serialised objects, one for each test identifier and every hour of the monitoring period. It should be noted that it is simple to vary this time granularity for other applications.

All applications then use the primitive database manager to read the required objects (i.e. records in the table) from the disk.

### 4. Network performance intermediate information

The network performance analysis process usually requires summaries for a specific analysis period. These include various statistical summaries related to the delay experienced by the test packets. Due to the large amount of data collected during the monitoring operation, calculation of these separate summaries can computationally be an extremely intensive process.

transmitted data table		
transmit time	time, $\mu$ s	packet identifier
36AF23C0	198723	231678
36AF23C1	200019	231679
36AF23C2	190011	231680
36AF23C3	195132	231681
36AF23C4	189891	231682
36AF23C5	210086	231683
36AF23C6	207562	231684
36AF23C7	196324	231685
36AF23C8	190981	231686

received data table				
transmit time	time, $\mu$ s	packet identifier	receive time	time, $\mu$ s
36AF23C0	198723	231678	36AF23C0	205848
36AF23C1	200019	231679	36AF23C1	207548
36AF23C3	195132	231681	36AF23C3	202119
36AF23C4	189891	231682	36AF23C4	196900
36AF23C5	210086	231683	36AF23C5	217079
36AF23C6	207562	231684	36AF23C6	214676
36AF23C6	207562	231684	36AF23C6	220040
36AF23C7	196324	231685	36AF23C7	203179
36AF23C8	190981	231686	36AF23C8	198194

Fig 5 Transmitted and received data tables.

transmit time	time, $\mu$ s	packet identifier	receive time	time, $\mu$ s	delay, $\mu$ s
36AF23C0	198723	231678	36AF23C0	205848	7125
36AF23C1	200019	231679	36AF23C1	207548	7529
36AF23C2	190011	231680	0	0	0
36AF23C3	195132	231681	36AF23C3	202119	6987
36AF23C4	189891	231682	36AF23C4	196900	7009
36AF23C5	210086	231683	36AF23C5	217079	6993
36AF23C6	207562	231684	36AF23C6	214676	7114
36AF23C6	207562	231684	36AF23C6	220040	12478
36AF23C7	196324	231685	36AF23C7	203179	6855
36AF23C8	190981	231686	36AF23C8	198194	7213

Fig 6 Pre-joined primitive data tables.

In order to enhance the efficiency of the network performance analysis operation, the database employs information reusability. The following sections explain the data structures used for the network performance intermediate information derived from primitive data.

#### 4.1 Delay distributions

The frequency distribution of the delay experienced by test packets during a specific analysis window (delay distribution) represents a reusable information structure. A frequency distribution is derived by generating a tabular grouping of primitive data and reporting the number of observations in each category. Frequency distributions place data in order so that visual analysis of the measurements can be performed. They also provide a convenient structure for many computations which fundamentally require an ordering of the data as an initial

part of their operation. Frequency distributions can be used to derive the following additive as well as non-additive summaries:

- mean,
- variance,
- number of test packets received within the specific analysis window,
- percentiles,
- mode.

Delay distributions cannot provide accurate minimum and maximum delay values within a specific analysis window. However, the class representative of the first non-zero class can be used as an approximate minimum delay value. Similarly the class representative of the last non-zero

class can be used as an approximate maximum delay value. In order to provide the required information with reasonable accuracy, suitably small classes or bins need to be used. The number of bins used will therefore relate to the accuracy of the information generated.

Delay distributions derived for smaller non-overlapping analysis windows can be added to provide a delay distribution for a larger analysis window provided that the delay distributions for the smaller analysis windows contain equal numbers of bins and bin sizes. Thus the required summaries can be derived at various levels of granularity without accessing the primitive data.

In the simplest case, these delay distributions are generated by choosing a maximum value of delay that a test packet be expected to experience. The bin size is a function of the maximum delay value and the number of bins required in the distribution. In an analysis window, there may exist test packets which experience delay greater than the maximum value specified. In this case, a delay distribution is derived for delay values lower than the specified maximum delay value. For delay values greater than the specified maximum, the highest value of delay and the number of test packets that experience delay greater than the specified maximum are stored.

Delay distributions are stored physically as serialised objects, one for every delay distribution. These objects also contain methods that derive appropriate summaries from the distribution data.

Other structures for intermediate information are possible. For example, sorted delays rather than delay distributions may be used.

#### 4.2 Packet-count objects

Packet-count summaries include the number of test packets:

- transmitted over the network,
- received by the receiving test station,
- lost during communication,
- duplicated during communication.

These summaries are also required for specific analysis windows within the analysis period.

Due to the additive nature of these summaries, adding the packet counts calculated earlier for smaller, non-overlapping analysis windows (Fig 7) could derive packet counts for large analysis windows.

time	transmitted	received	lost	duplicated
00:00	3600	3591	9	0
	+	+	+	+
03:00	3540	3523	21	4
06:00	3600	3590	11	1
	+	+	+	+
09:00	3600	3599	1	0
12:00	3100	3100	0	0
	+	+	+	+
15:00	3600	3600	0	0
18:00	2500	2501	0	1
	+	+	+	+
21:00	3600	3600	0	0



time	transmitted	received	lost	duplicated
00:00	7140	7114	30	4
06:00	7200	7189	12	1
12:00	6700	6700	0	0
18:00	6100	6101	0	1

Fig 7 Additivity of packet count summaries.

Packet-count summaries are managed in hash tables, one for each test identifier and analysis window size. The test identifier and the analysis window size are treated as implicit category attributes [11] and are not used for the identification of statistical entities within the statistical table.

### 5. A generic architecture for network performance preprocessing

This section describes a generic object-oriented architecture for an information preprocessor based on the data structures and processing techniques described in the previous sections.

The information preprocessor receives a query specification from a client or a higher level object. In response to this query, the preprocessor returns the appropriate intermediate information objects to the client object. The client object can then derive the required information from these preprocessed intermediate information objects (Fig 8).

The query specification includes the start and stop times for analysis periods, the test identifier and the analysis window size for which the information is to be processed. The preprocessor uses a query decomposer object to decompose the original query into its component query specifications. Each component query specification has the analysis period equal to a specific analysis window within the original analysis period.

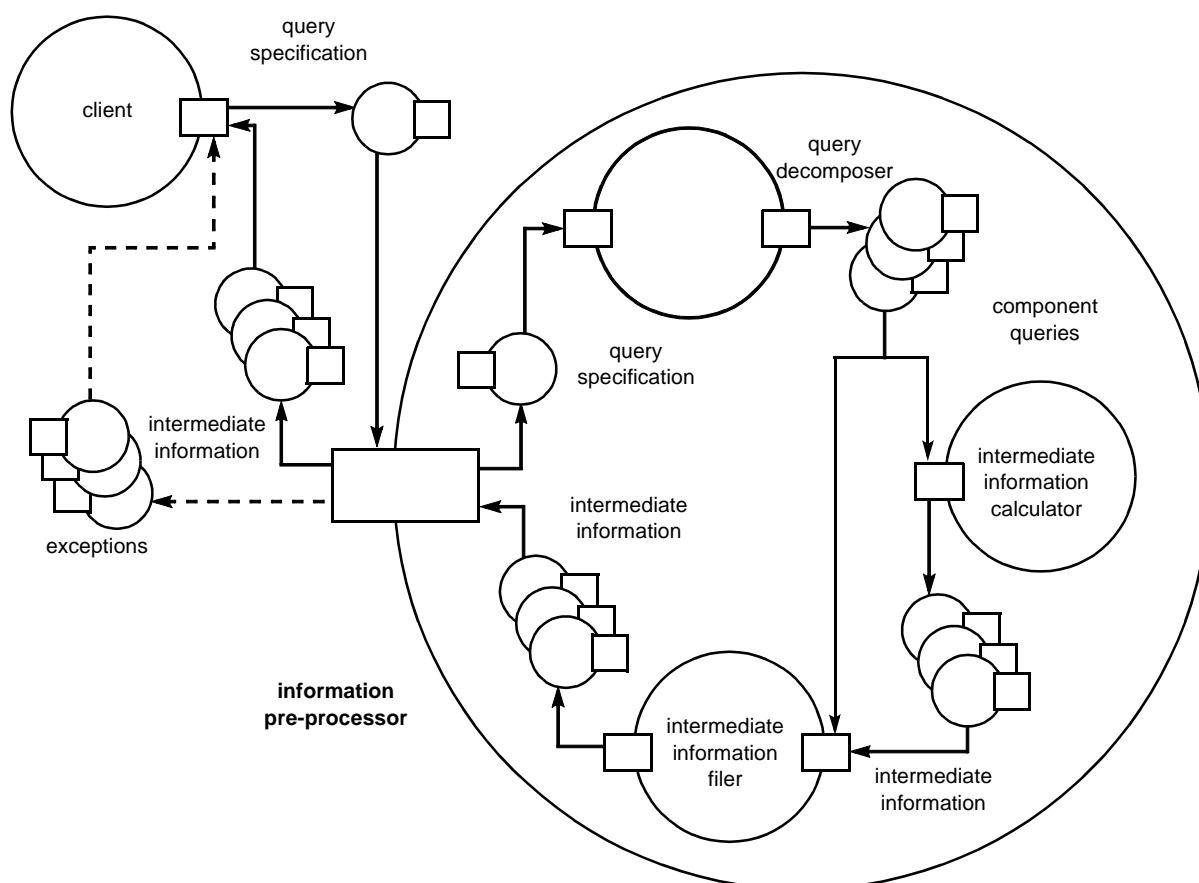


Fig 8 Information preprocessor.

Each component query is passed to an intermediate information filer object. This object manages the storage and retrieval of preprocessed intermediate information objects. If objects corresponding to the component queries exist, these are retrieved and returned to the client object. Alternatively, if an intermediate information object does not exist for a specific component query, that component query is passed to an intermediate information calculator object. This object attempts to construct the required intermediate information elements from the finest granularity intermediate information elements or primitive data. All intermediate information objects constructed by the intermediate information calculator are passed to the intermediate information filer, which stores them for subsequent reuse.

The information preprocessor uses the initial query specification to initialise different components. Intermediate information objects corresponding to successive analysis windows are pulled by successively triggering an appropriate method in the information preprocessor. The client object can then extract the relevant information from these intermediate information objects.

The intermediate information calculator object uses a query decomposer object to decompose each component query (previously generated from the original query

specification) into query specifications requesting intermediate information for specific one-hour periods within the analysis window. These are called granular queries and are used by an intermediate information filer object to retrieve the relevant preprocessed granular intermediate information from files on disk. All granular intermediate information objects corresponding to these granular queries are passed to an intermediate information combiner object. This object combines the granular intermediate information elements to provide the intermediate information object at the requested level of granularity (Fig 9).

## 6. Database performance characteristics

In this section, the performance of an information system using the information preprocessor (referred to as the delay distribution system) is compared with that of an information system calculating the same user-requested information from the primitive data each time a query is submitted to it (this latter system is referred to as the Primitive Database). These systems have been programmed in Java and operate under the Solaris operating system on a Sun Sparc20 computer.

The data for these tests was collected by transmitting one test packet per second on a particular network link.

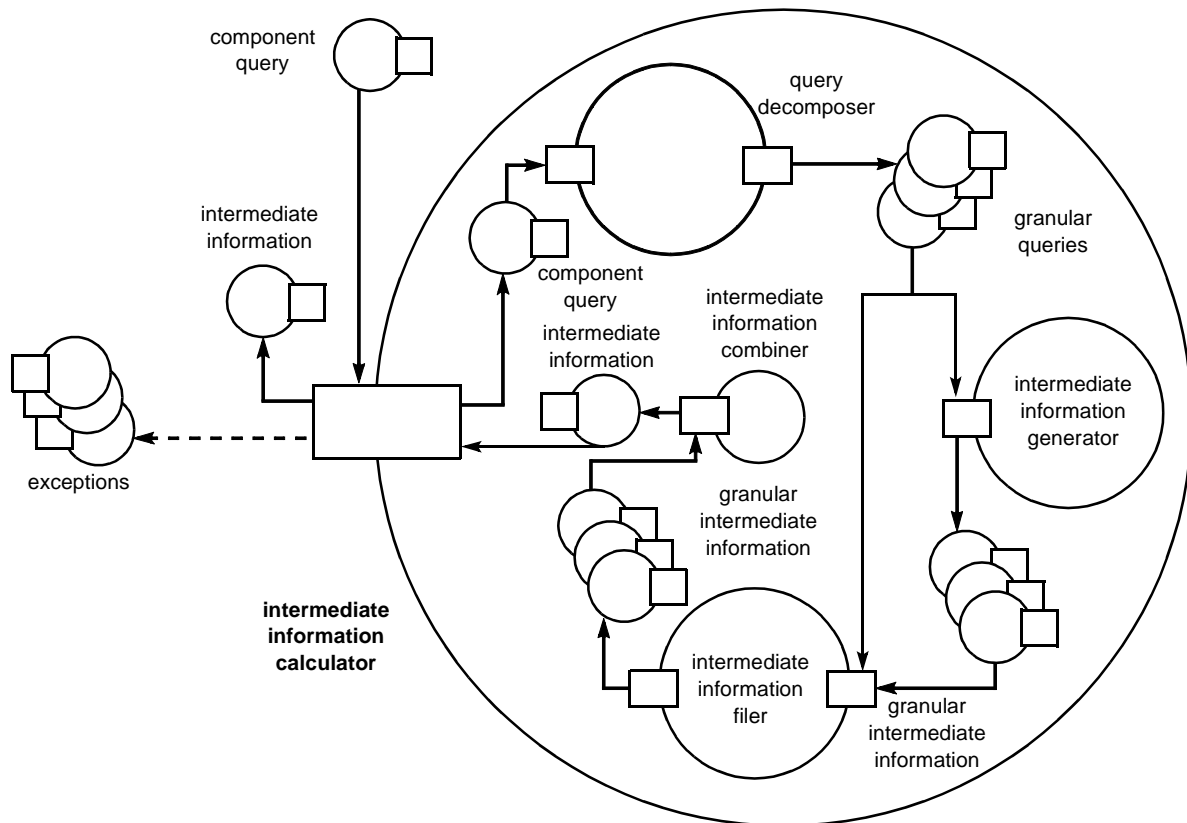


Fig 9 Intermediate information calculator.

Thus in one hour of monitoring, 3600 test packets were transmitted by the transmitting test station.

The performance of the delay distribution system was measured when:

- no intermediate information was available,
- intermediate information was available only at the finest level of granularity,
- intermediate information objects at the required levels of granularity were available.

The delay distributions (intermediate information) for these tests were derived by assuming that the maximum delay that a packet may experience was always less than 10 ms. Each delay distribution had 2000 bins. Two different types of queries were then submitted to these information systems.

The first query requested the average packet delay values for the successive 5% fastest test packets during a specific analysis window. To process this query, the Primitive Database extracts the appropriate packet-delay values from the primitive database, sorts them in order and then derives the required information. On the other hand the delay distribution system attempts to locate the preprocessed delay distribution object. If this object does

not exist, it attempts to derive it by combining appropriate finest granularity delay distribution objects. All the finest granularity delay distribution objects that have not been preprocessed need to be derived from the primitive data. Once the required delay distribution object has been accessed or generated, the delay distribution system processes it to provide the required information.

Figure 10 shows that the performance of the delay distribution system is higher (low response times) even for the worst-case scenario, i.e. delay distribution objects at any level of granularity are not available. This is due to the efficient sort algorithm used. Moreover, when delay distributions at the required level of granularity are available, the performance of the delay distribution system is constant and does not depend upon the analysis window size. It should be noted, however, that, as the size of the primitive data set grows with increase in the analysis window size, the performance of the Primitive Database deteriorates significantly.

The second query requested the average packet delay value for all test packets transmitted and received within a specific analysis window. In this case, the Primitive Database simply accesses the required primitive delay values and calculates the mean delay for the specific analysis window. The operation of the delay distribution system is the same as in the previous case.

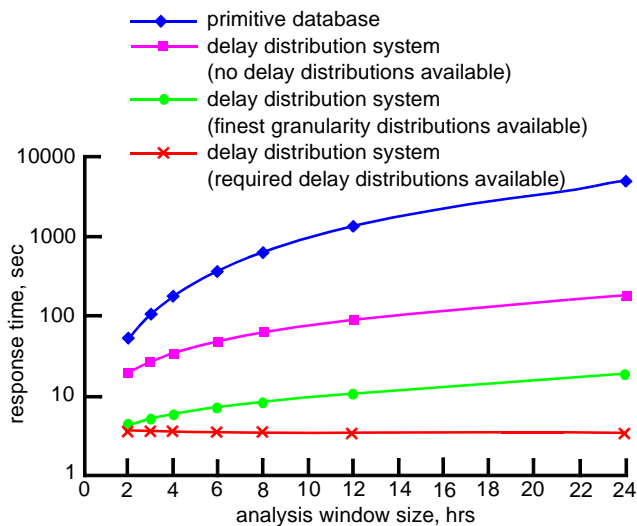


Fig 10 Performance comparison (query: average delay of successive 5% fastest test packets).

The performance of the Primitive Database is slightly higher (i.e. lower response time) than the performance of the delay distribution system for the worst case scenario, i.e. delay distribution objects at any level of granularity are not available. This is because sorting of primitive data values is not required to calculate the mean. Moreover, there is an overhead in generating and storing the delay distributions at the finest granularity. However, if the delay distribution system reuses these preprocessed finer granularity delay distributions, its performance is enhanced significantly (Fig 11).

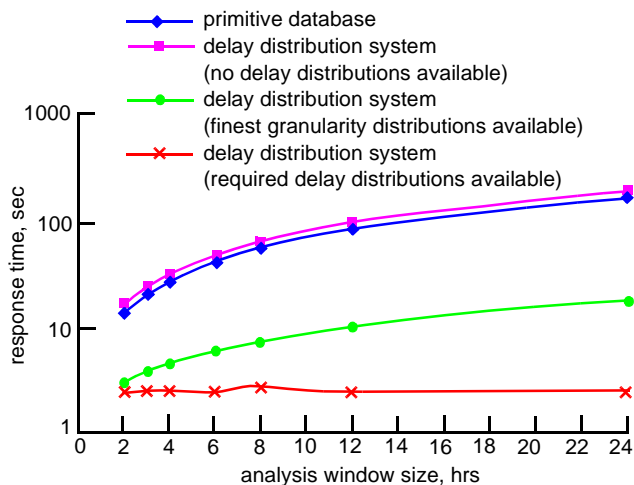


Fig 11 Performance comparison (query: mean delay value).

Delay distributions can be derived for a number of different bin sizes. The choice of the bin size has implications on the performance of the information system using these delay distributions as well as the accuracy of the information derived from them. Figure 12 shows the performance of the delay distribution system using delay

distributions of 2000 bins with respect to its performance when it is using delay distributions of 500 bins.

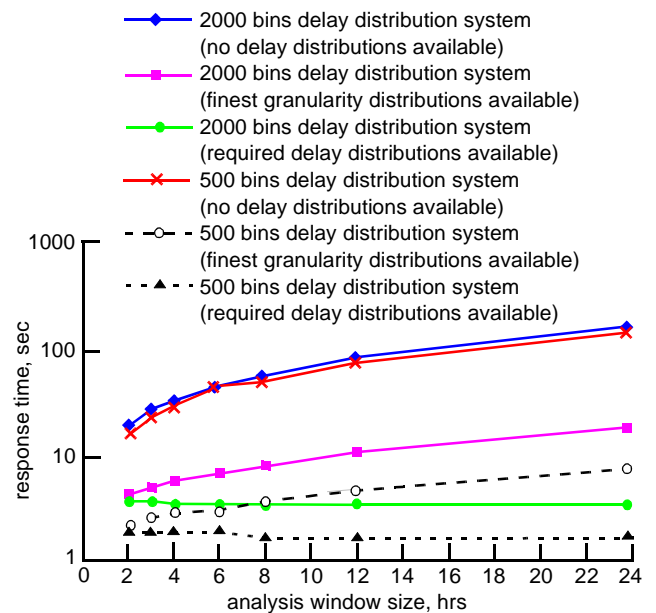


Fig 12 Performance of delay distribution system for different bin sizes.

The response time is shorter if a smaller bin size is used. However, the accuracy of the results is also reduced. A compromise between response speed and accuracy could therefore be selected.

## 7. Network performance information service

Network performance information may need to be disseminated to a number of different human and computer clients. The information requirements of these clients may vary significantly. The information service must therefore be able to fulfil diverse information requirements efficiently. This section describes a network performance information server developed using Java's Remote Method Invocation (RMI) mechanism (Fig 13).

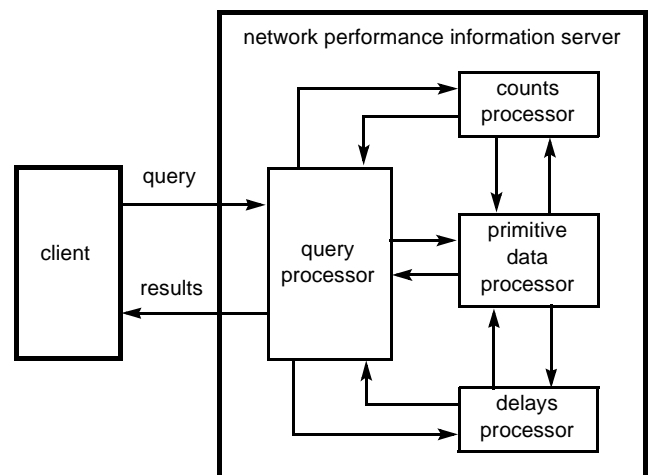


Fig 13 Network performance information server.



The server can provide three types of information element to a client:

- primitive data as pages containing transmitted and received data packets,
- delay distributions and packet counts derived at a specific level of granularity, i.e. intermediate information,
- processed summaries derived at a specific level of granularity, e.g. the average delay values for the successive 5% fastest test packets along with the corresponding packet counts.

Thus clients with minimum resources (thin clients) can acquire fully processed summaries from the server, whereas clients with substantial resources (fat clients) can request the server to provide primitive data or intermediate information as available. The resources of these clients may be exploited by caching the retrieved data and then deriving the required summaries locally.

The query processor receives all queries posted to the server. It determines the type of information requested by the client. If the request is for primitive data, the query processor triggers the primitive data processor to provide the required pages. If the client requests intermediate information, the query processor triggers the counts processor and delays processor to provide the required objects. If any intermediate information has not been preprocessed, the counts and the delays processors request the primitive data processor to provide the required primitive data and then these are processed to provide the required intermediate information objects. Requests for processed summaries are fulfilled by deriving the required summaries from the respective intermediate information.

The query manager also formats the queried information and data for communication to the client.

## 8. Conclusions

Network performance monitoring activities generate very large data sets. These need to be summarised appropriately in order to efficiently analyse the performance of the monitored networks. User information requirements vary significantly, however, as different users request different types of summaries at varying levels of granularity and analysis periods. For this reason, preprocessing some selected summaries may prove to be a short-sighted decision.

It may therefore be desirable to preprocess the data so that different user information requirements can be fulfilled efficiently.

Reusing packet-loss and duplication summaries is relatively simple due to their additive nature. Packet losses and duplications counted at finer levels of granularity may simply be added to provide similar information at coarser levels of granularity. However, some valuable summaries related to the delay characteristics of the network are not additive in nature and cannot be decomposed into additive components, e.g. percentiles. Reusing these summaries is therefore not possible.

Frequency distributions of packet-delay values, however, do represent a reusable data structure, which is additive; a wide variety of additive and non-additive summaries can then be derived from such distributions. Distributions derived at finer levels of granularity can be combined to provide a frequency distribution at a coarser level of granularity. As access to the primitive database is reduced significantly, there is a significant enhancement in the performance of the information processing and analysis applications.

This paper also proposes a generic object-oriented architecture for database components that use such intermediate information elements to fulfil different information requirements.

## References

- 1 Siddiqui M H: 'Performance measurement methodology for integrated services networks'. PhD Thesis, Loughborough University of Technology, UK (1989).
- 2 Siddiqui M H, Parish D J and Adams C J: 'Performance of local and remote bridge components in Project Unison — 50 Mbit/s ATM network', Proceedings of the 1989 Singapore International Conference on Networks, pp 1—6 (July 1989).
- 3 Phillips I, Parish D J and Rodgers C: 'Performance measurements of the SMDS', IEE Colloquium on SMDS, London (October 1995).
- 4 Phillips I, Tunnicliffe M J, Parish D J and Rodgers C: 'On the monitoring and measurement of quality of service of SuperJanet', 13th Teletraffic Symposium, Strathclyde (March 1996).
- 5 Jain R: 'The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modelling', John Wiley & Sons Inc (1991).
- 6 Brooks R, Blattner M, Pawlak Z and Barrett E: 'Using partitioned databases for statistical data analysis', Proceedings of the AFIPS National Computer Conference, pp 453—457 (1981).
- 7 Mumick I S, Quass D and Mumick B S: 'Maintenance of data cubes and summary tables in a warehouse', Proceedings of the ACM SIGMOD Conference, Tucson, Arizona (May 1997).
- 8 Lenz H and Shoshani A: 'Summarizability in OLAP and statistical databases', Proceedings of the 9th International Conference on Scientific and Statistical Database Management, Olympia, Washington (August 1997).
- 9 Heller P and Roberts S: 'Java 1.1 Developer's handbook', SYBEX Inc (1997).

10 Phillips I, Bashir O and Parish D: 'Performance monitoring of networks: architecture, operation and dissemination', Presented at the BT URI (University Research Initiative) Workshop, University College, London (April 1997).

11 Rafanelli M, Bezenchek A and Tininini L: 'The aggregate data problem: a system for their definition and management', SIGMOD Record Journal (December 1996).

---

Omar Bashir received his BE in Avionics Engineering from NED University of Engineering and Technology in December 1989. He received his Masters in Information Technology and PhD from Loughborough University in 1995 and 1998 respectively.



including BT, EPSRC and RACE (Europe).

Iain Phillips graduated from Manchester University with BSc and PhD degrees in Computer Science. From 1992 he has worked as a Research Assistant and later a Research Fellow in the Department of Electronic and Electrical Engineering at Loughborough University. His research areas include all aspects of multiservice networks, in particular performance measurement, with other areas including video error concealment, system integration and specialist ATM networks, in particular telemetry and video surveillance. This work has been funded from a variety of sources



management of multiservice networks using SuperJanet. He heads the HSN) at Loughborough University.

David Parish holds BSc and PhD degrees from the University of Liverpool. He has worked as a Scientific Officer at the UKAEA Culham Laboratory and as a Demonstrator at the University of Liverpool. In 1983 he joined the Department of Electronic and Electrical Engineering at Loughborough University as a Lecturer, later becoming Senior Lecturer and then Reader in Multiservice Networks. His research interests concern the management, operation, monitoring and application of high-performance networks. Specifically, he leads Loughborough University's input to the BT-funded research programme into the



John Adams joined BT in 1971 and worked initially on traffic management issues and, later on, worked on part of the System X digital switch development programme.

From 1983 he undertook some fundamental research on multiservice local area networks and went on to invent the Orwell Ring LAN protocol for carrying integrated voice, video and data on a shared slotted ring network.

The natural convergence of these ideas with the budding developments in asynchronous transfer mode (ATM) led him to join the

Broadband and Data Networks unit at BT Laboratories, working primarily on ATM traffic control issues and ATM standards.

Currently, he is an Engineering Advisor with a focus on broadband ATM or IP traffic design and associated standards.

He is a Chartered Mathematician and co-author (and, in one case, the editor) of two books on ATM and LAN issues. He is also the author of numerous papers on ATM.



Tim Spencer joined BT Laboratories in 1989 having graduated from the University of Southampton with a BSc in Mathematics.

Initially he worked in the transwitching section on network modelling and simulation studies for BT's international backhaul network.

In 1995 he moved to the Engineering Advisors Office within Broadband and Data Networks, where he led activities on network performance management and provided expertise to other areas of BT.

Since September 1998 he has been in the Intelligent Solutions unit developing computing telephony applications and services for corporate intranets and small businesses.