Traffic Performance Specification and Modelling in the Intelligent Network

Philip Richards, Romas Armolavicius, Rod Rogers
Bell-Northern Research
P.O. Box 3511, Station C
Ottawa, Canada, K1Y 4H7

Surinder Chhura
Bell Canada
220 Laurier Ave. West
Ottawa, Canada, K1P 5Z9

The implementation of Intelligent Networks (IN) will uncouple service design and deployment from a traditional strong dependence on specific switching and transport systems. This paper argues that an inevitable consequence is that new approaches to service modelling and performance analysis will be required. The paper begins with a brief description of the enabling architectural principles that underlie the IN concept together with the factors that motivated their development; a proposal for new modelling approaches is then made within the context of the new IN architecture.

1. Introduction

The ability to rapidly trial and deploy new services is vital to the dynamic business needs of both service providers and their customers. Historically, service velocity, or the rate of introduction of new services by service providers, has been low. This was largely due to the structure of most telecommunication networks, insofar as:

- Originally, only point-to-point voice services were supported, using a simple, universal mode of interaction with users (dialing strings of digits) as a means of service control. Since the network was not designed to support complex, changing services, its structure and embedded mode of operation presents considerable inertia to change.

- The introduction of Stored Program Control enabled the development of sophisticated node-based services. However, the lack of application (service) level interworking standards among various vendors' node products has hindered the emergence of complex network based services.

- The process of service definition, development, implementation, testing, and deployment is a long, error-prone, and arduous one, not embarked upon lightly by service providers without the potential of significant revenues per service developed. Service velocity has been further retarded by a lack of capabilities to trial services on a small scale, quickly, and at low cost.

- The corporate service user community has been largely excluded from service design. Since these users are likely sources of innovative ideas for new and better services, this lack of participation may have hindered new service development.

The notion of Intelligent Network (IN) was introduced in the 1980's as a means to both increase service velocity and allow service customization by a much wider community, specifically including telephone operating companies. There are two main requirements that dominate the design of the Intelligent Network. These are:

Service and technology independence:
This refers to the ability to add new IN services to the network without making changes to the underlying capabilities of the switching and transport systems, and without being restricted to deployment over particular equipments or technologies.

Interworking with existing switch based services:
This allows a graceful evolution to the Intelligent Network while retaining much of the existing service base.

A number of architectural proposals have been made through standards bodies for the repartitioning and extension of network intelligence in accordance with the above requirements. For the purpose of this paper, a simplified architectural view is sufficient (based on the standardization efforts of Study Group XI of the CCITT [1]).

A key notion in this view is that new services may be constructed from a finite set of reusable Service
Independent Building-blocks or SIB's.
Examples of functions provided by SIB's include:

- simple database look-ups,
- complex conditional database operations (conditioned on such parameters as the time of day, the day of week, the date, the geographical area of call origination, etc.),
- requests for external data (from a remote database, for example), and
- the launch of a "prompt-and-collect" sequence with either the originating or terminating party to obtain additional information (for example, a personal identification number for authorization purposes).

The functionality described above is service-independent in that it is designed to be reused across all IN services. Each service also has functionality that is specific to that service, which defines its operation and external appearance. This service-specific functionality comprises:

- service logic which is responsible for invoking and controlling the sequence of execution of the SIB's employed by the service,
- data which is used by the service logic (for example, customized announcements or prompts passed as parameters to SIB's responsible for delivering them to service users),
- the activation/deactivation status of the triggers which are used to invoke service logic.

The development, testing, and deployment of service-specific logic and data are the responsibility of the Service Creation Environment (SCE) and the Service Management System (SMS). These allow rapid service creation, delivery, and customization by service providers in the following manner:

- A service designer uses the SCE to develop the logic for a new IN service: this logic specifies the appropriate SIB's to be invoked, in the required order to execute the intent of the service.
- After functional and performance testing is complete (using tools within the SCE), the service logic is submitted to the SMS. When the service is to be launched, appropriate logic/data combinations are downloaded by the SMS to the various product platforms that have been selected to support the service. When activated by a service transaction, the logic/data is interpreted and executed by a set of service-independent generic programs that reside on the platforms.

2. Traffic Performance Implications

The independence of service design and platform implementation will require a corresponding separation of responsibilities in traffic performance specification and modelling. In particular, end-to-end service performance seen by the customer will be a function of three distinct responsibilities:

Service Performance Verification:
This would be the responsibility of the service provider. It will be assumed to be carried out as an integral part of the Service Creation process.

Specification of Platform Performance:
Performance specifications would remain the responsibility of appropriate industry or standards bodies working with both manufacturers and service providers. These specifications would play a role analogous to call processing delay specifications for today's switching machines but would be service-independent. Compliance with the specifications would be the responsibility of platform equipment manufacturers.

Platform Provisioning:
This would be the responsibility of network and systems engineering groups.

The balance of this paper considers how each of the preceding areas should be addressed, given that the overall motivation is consistent, good performance seen by all end-users.

3. Service Performance Verification

The traditional service development process requires the "custom-crafting" of service-specific models to ensure that a service design conforms to performance standards and does not adversely affect the operation of other services sharing the same resources. Consequently, service performance analysis today is:

- An informal, but integral, part of the service development process, and
- Vertically integrated, based on a detailed knowledge of both the service logic and of the architecture and operation of the supporting platforms.

In the future IN environment, service performance analysis must be available to service designers from the SCE, imposing the following requirements:

- Service performance analysis must be formal and based on the same models used to express service designs within the SCE.
- The SCE must provide a platform-independent performance characterization of the service being designed. The SCE would assume that the underlying physical platforms comply with their performance specifications and are properly provisioned to guarantee specified objectives. As
these objectives are required to be platform independent, they should be based on a specification of the performance required from each SIB. These requirements can then be used to analyze the performance of the IN service itself by using methods such as those described in [2].

In order to understand how these requirements might be met, we need to consider the nature of IN services in more detail. Specifically, IN services (as presented to the service designer by the SCE) exhibit a structure or architecture and a behavior. These are discussed in sections 3.1 and 3.2; sections 3.3 and 3.4 then apply these concepts to consider product independent performance characterization and the setting of service performance objectives.

3.1 Service Structure

The service structure describes the functional components that comprise the service together with their communication relationships. Figure 1 depicts a typical structural description based on [1].

![Figure 1](image-url)

The structural description presented in the diagram above employs three major IN functional components:

- **Call Control Function (CCF):**
  This is responsible for call and connection setup and control (as well as the control of other switch-based resources), and typically contains "trigger points" used to invoke and interact with IN service logic.

- **Service Switching Function (SSF):**
  This is responsible for providing a well-defined, service-independent interface between the Call Control Function and the Service Control Function (below).

- **Service Control Function (SCF):**
  This interprets and executes the logic and data which implement a given IN service.

The CCF/SSF is service-independent in the sense that it requires no "knowledge" of customer and service-specific data and logic. Conversely, the SCF requires no "knowledge" of switch-based call-processing and switch-based services nor of call-state information known to the CCF/SSF.

Each of the functional components described above has a role in determining the overall performance of the IN service. It should be noted that the user process is included in the structural description shown in Figure 1; in particular, it must be included in the specification of how the IN service is deployed as communication with the user's terminal is performance affecting.

3.2 Service Behavior

A specification of the possible internal states of each functional component together with the conditions causing a change of state within the context of a given IN service are critical to performance analysis. A variety of representational techniques are available for capturing such behavioral descriptions (extended finite state machines for example). The performance analysis mechanisms within the SCE must be able to construct appropriate performance models for services expressed in the representational scheme that the SCE supports. The behavioral specification is needed to describe the following:

- **How Functional Components interact with each other:**

  In particular, the state transitions that describe the interactions between the user and the service provide a set of externally-perceivable events that can serve as the basis for performance objectives and measurements. For example, Figure 2 below shows the pair of events:

  - finish dialing digits, and
  - receive audible ringing or busy tone.

  These two events define the traditional call-setup-delay performance measurement.

![Figure 2](image-url)
description of the generic states, state transitions, and IN service triggers that comprise call processing at an end office and is the basis for creating platform independent IN services. The SCE must provide access to such models (or at least the ability to define them) for incorporation into the performance model of an IN service.

- **How Functional Components use Resources:**

Typical hardware resources are service circuits (such as the tone receivers shown in Figure 3 below); software resources include communication services (common channel signalling), database management services, and connection management services. In the platform independent representation that we have been considering, all of these resource usages are expressed as activations of appropriate SIB's.

![Figure 3](image_url)

- **How each Functional Component's processing tasks are organized:**

This includes control constructs such as conditional branches as well as protective mechanisms such as timeouts.

The preceding discussion has stressed that service performance analysis must be formal and based on the same models used to express service designs within the SCE.

![Diagram](image_url)

3.3 Platform Independent Performance Characterization

A formal model which captures the structure and functional behavior of an IN service is necessary but not sufficient for the performance analysis of that service. It has already been mentioned that the service model includes descriptions of how resources are used by the IN service. To analyze the performance of the service, the temporal behavior of these resources must be specified: for example, in Figure 3, the duration between the pair of events:

- request receiver attachment
- receiver attached

must be characterized. This applies to all resources that can be accessed by SIB activations. At this point an obstacle arises, in that resource performance depends on factors not normally considered within the purview of the SCE, such as:

- Global demand for resources arising from either the concurrent execution of multiple instances of the service under consideration, or from other services which make use of the same resources,

- Proprietary implementations of resources (typically within a single platform such as an end office switch),

- The impact of platform configuration or service deployment decisions (for example, messages exchanged by two IN functional components will experience greater delays in a network environment based on common channel signalling than they would if both of the functional components were mapped to a single processor), and

- Resources that are shared by concurrently executing services form an implicit coupling point among these services. The monopolization of any resource by any one service will affect the performance of other IN services needing the same resource.

In this paper, the ensemble of such performance-affecting details will be called the (performance analysis) environment.

In order to deal with the preceding issues and to provide a basis for a workable framework for performance analysis within the SCE we need a simple way to characterize the influence of the environment. To this end we propose the following assumptions:

**Assumption 1:**

The main function of SCE based IN performance analysis is to ensure that the service users' perception of performance levels is within specified limits.

The methodological implications are that:

- The events needed to characterize the user's perception of performance must be included in the service model.

- The service designer may need to make design decisions on the basis of his performance analysis. The service performance model must contain sufficient design detail to facilitate such decisions.
- Performance objectives for interactions across the user-service boundary must be available (this will be discussed in more detail in the next section).

In summary then, the stress is on the detailed analysis of a single instance of the IN service as viewed by the user.

Assumption 2: Implementation details below the functional interface are not visible from the SCE.

Consequently, every resource visible at the functional interface (as expressed in terms of SIB's) must have an associated performance model. This model should have a number of characteristics:

Simplicity:

Consideration must be given to the fact that the resource performance model will be used to construct a performance model of the complete IN service and therefore must be as simple as is practical. We postulate that (conceptually at least) the model takes the form of pair \((C, P)\) where:

- \(C\) is a qualifying condition, such as "average busy season network busy hour" which describes the prevailing conditions in the IN service's environment.

- \(P\) is the performance representation expressed as a probability distribution, such as \(P(t) = \text{Prob}(\text{Delay} < t)\) for a given state \(C\) of the environment.

The distribution \(P\) must have a form that is amenable to analysis. To this end, \(P\) might be chosen from the class of phase-type distributions (see [3]). It is well known that this class is sufficiently rich that any "reasonable" delay distribution may be approximated as accurately as needed: in particular, exponential distributions are of phase-type. The advantage of using this class of distributions is that they allow the standard theory of Markov processes to be applied directly.

Coverage:

A range of resource models \((C, P)\) could be provided to span any desired range of analyses, for example:

- **Worst case** (each resource is assumed to perform at the limit of its acceptable performance)

- **Nominal case** (each resource is assumed to perform at some nominal point within its performance envelope)

Each of these categories could also be further qualified by the state of the operating environment (for instance, \(C = \"high day busy hour\"\) conditions).

It is clear from the preceding that there is a proliferation of cases, each with its corresponding model. A potential method for reducing the number of cases may be to use extreme value engineering.

The preceding analysis is implicitly predicated on the assumptions that follow.

Assumption 3: The introduction of any service into the network does not affect the resource performance models.

This is a critical assumption. It states that by appropriately engineering the physical platforms, the performance required of resources will be provided. It should be noted that this assumption does not state that there will be no impact of deploying the service, as in particular, capacity will be consumed. It does state however that (at least to "first order"):

- service coupling through common resource pools, and

- demand for resources arising from specific services can be hidden below the mechanisms (SIB's) used to access resources. We then are allowed to use the resource models with impunity, leaving it to the platform design and provisioning processes to ensure that the required performance levels are met.

The performance characteristics of the resource models will in general depend on the configuration of the underlying platforms. We assume that a standard collection of representative such configurations can be chosen to serve as reference models to simplify the analysis process. Specifically, we propose the following:

Assumption 4: The impact of deployment decisions can be captured through the use of reference configurations.

The situation faced by the performance analyst is as shown in Figure 4:

![Figure 4](https://example.com/figure4.png)
select the deployment alternatives that are appropriate for the desired analysis. Assumption 4 proposes that a number of standard reference configurations of "generic platforms" such as switches, Service Control Points (SCP's), etc. be made available. The service designer, by mapping the functional components of an IN service onto a configuration as shown in Figure 5, directs the SCE resident performance analysis tools to select matching resource performance models. The subsequent analysis of the IN service will then reflect the deployment decisions taken by the designer.

For example, in Figure 5, the delay between the message emission and reception events shown depends on the nature of the communication relationship between the physical platforms onto which the CCF/SSF and SCF have been deployed. In an integrated End Office - SCP configuration, message transit times will be less than in the case shown in Figure 5, where a standalone SCP and a CCS7 STP are involved.

![Figure 5](image)

3.4 Service Performance Objectives

As in today's environment, performance standards will be necessary at the user-service network boundary to ensure that services conform to levels of performance to which users are accustomed. Classical examples of such standards are dial-tone-delay, and call-setup delay. Although this pair of measures does not lose its importance in an IN setting, the situation is becoming more complex for the following reasons:

- **Terminals and access protocols are becoming more sophisticated:**

  With the increasing penetration of ISDN services and functional signalling, there is a migration of processing to the terminal. Accordingly, the performance perceived by the user is a mixture of the performance of the network and of the performance of the hardware and software of the terminal. Objectives can reasonably be imposed only on the network based performance component: consequently, the corresponding performance objectives are no longer "pure" in the sense of being directly perceivable.

- **IN services can have arbitrarily complex interactions with the service user:**

  IN services are open-ended in the sense that they permit the inclusion of very complex data processing, searching, and routing tasks while the user is waiting for a response. The risk of excessive delays is exacerbated by the inherently distributed nature of IN services, potentially subjecting users to long transmission as well as processing delays.

  To meet the challenge of measuring, predicting, and controlling IN service delays, it may be necessary to specify objectives for distinct classes of user-service interactions. By grouping interactions at the user-service boundary into classes based on inherent processing difficulty, each class can be given a different delay objective (Users will tolerate longer delays for tasks they perceive to be more complex. Also, user acceptance can be enhanced by reinforcing the perception of the complex nature of the task by supplying appropriate feedback in the form of auditory or visual progress messages). A small number of classes should suffice, such as:

  - **Low complexity tasks:** mean delay < 1 sec. (This would cover routine interactions such as initial network access as measured by dial-tone-delay on digital switching machines)
  - **Moderate complexity tasks:** mean delay 1-3 sec. (This would cover interactions such as call-setup using common channel signalling as well as invocations of simple IN services)
  - **High complexity tasks:** mean delay > 3 sec. (This would cover complex interactions involving database operations, connection management, or routing. An IN service example in this category might be network Automatic Call Distribution.)

It is beyond the scope of this paper to recommend
specific interaction classes, objectives, or service classification criteria; the preceding are presented for expository purposes only.

4. Specification of Platform Performance

Platform performance will need to be expressed in terms of its response time to requests for access to, and manipulation of logical resources, for example:

- the expected time to set up a channel,
- the percentile of delay tolerable to connect a tone receiver, and
- the expected time to perform a database look-up.

The requests would take the form of operations across the functional interface and would be the basis for the analysis of services as described in the preceding section. Once the functional interface and its constituent primitive operations (SIB’s) are defined, each operation will have to have one or more performance attributes (as well as appropriate numerical targets) associated with it.

Reference service scenarios will be needed to allow service providers and independent monitoring agencies to check for platform compliance under controlled conditions. These reference scenarios would take the form of standard service models, associated demand scenarios, and typical physical network configurations. As the concept of a reference model was discussed in detail in section 3.3 (in the context of platform independent performance characterization) it will not be pursued further here: by analogy to that earlier discussion, the criteria for reference model selection would include:

- Realism: The models should represent realistic services.
- Coverage: The models should, collectively, characterize all aspects of a platform’s performance.
- Simplicity: The models should be no more complex than needed to satisfy the two preceding criteria.

5. Platform Provisioning

The current practice of using detailed service-specific forecasts to provision network resources will not be applicable in the context of an IN platform. In an environment where new services are being developed and customized to suit individual and telephone operating company needs, there is likely to be little, if any, service specific historical data that can be used to generate accurate forecasts for resource requirements. Furthermore, due to the fact that IN offers the capability for increased service velocity, the number and type of services in operation is likely to change over the period of the provisioning interval.

To reflect the preceding reasoning and to remain consistent with the IN architecture, a service independent provisioning methodology is required. In fact, we make the following assumption:

Assumption 5 Knowledge of how individual IN services utilize resources is not available to the platform provisioner. Only aggregate utilization data are available.

Given this assumption, the platform provisioner must adopt a resource oriented (as opposed to service oriented) provisioning process. In a resource oriented approach, the platform’s provisionable entities must be grouped into service independent resource pools. These pools would typically represent entities such as processing elements, files/databases, service circuits (such as receivers, tones, and announcements), and common channel signalling links. Having partitioned the platform into resource pools, each pool must be assigned individual performance objectives (such as response time) so that the overall platform performance objectives can be satisfied. The provisioning process therefore becomes one of ensuring that each resource pool has sufficient capacity to satisfy its performance objective under the aggregate load generated by the services that use that particular resource. A more detailed exposition of this approach appears in [4].

The provisioning process will likely be made up of two distinct sub-processes: one responsible for planning orderly capacity changes; the other responsible for demand servicing. These are discussed in the next two sections.

5.1 Pool Capacity Initialization

The objective of the pool initialization process is to make an initial estimate of the capacity requirements for each resource pool over the duration of a provisioning interval (say 6 months or a year). In accordance with Assumption 5 above, this process should not rely on service specific forecasts for resource requirements. Instead, statistical models and forecasting techniques, using both historical demand trends and knowledge of future events (as identified by the planning process), should be used to predict the aggregate, service independent, demand levels for each resource pool on the platform.

There are a number of techniques that can be considered for generating these demand forecasts. For example, if the underlying dynamics of the demand process are understood (e.g growth and periodicity), and a sufficient amount of historical data has been collected, then techniques such as Kalman filtering may be used to generate demand forecasts.
These forecasts must provide a statistical distribution for the random demand $D_t$ that will be offered to the pool at a future time $\tau$. If it is known that a provisioned resource pool capacity $C$ delivers a level $S$ of resource performance for a given demand $D$ by virtue of the relationship

$$S = \omega(D; C)$$

then a knowledge of the future demand distribution $D$ allows the resource pool capacity $C_{\alpha}$ to be found such that

$$\text{Prob}(\omega(D; C_{\alpha}) > S_{\alpha}) = \alpha.$$ 

Here $\tau$ represents the duration of the provisioning interval, and $\alpha$ represents the acceptable probability of violating the resource pool's stated performance objective $S_{\alpha}$ at time $\tau$.

5.2 Pool Servicing

In order to maintain a desired level of performance throughout the provisioning interval, actual resource utilization must be tracked and new forecasts continuously generated to identify any trends toward higher utilization levels than originally predicted during the initialization process. The detection of any such trend would trigger a recommendation for either capacity relief or reconfiguration of the platform resources. This process would make use of the same forecasting techniques as discussed above. However, in this case $D_t$ would be continuously updated on the basis of observed utilization levels, and the servicing process will be searching for instances where

$$\text{Prob}(D_t > C_{\alpha}) > \beta \quad \text{where} \quad 0 < t < T$$

and where $\beta$ represents the acceptable probability of violating the resource pool's stated performance objective $S_{\alpha}$ at any time $t$ within the servicing interval (of length $T$). Note that in this case $C_{\alpha}$ (determined as described in the previous section) now represents the capacity of the pool in its current state and $T$ represents the lead time required to provision additional equipment for a given type of resource pool.

6.0 Summary

This paper has presented arguments for new approaches to service modelling and performance analysis in the context of Intelligent Networks. It has described how the IN Architecture can be used to control the complexity of performance analysis by uncoupling the three functions of service design and deployment, platform performance specification, and platform provisioning.

References

[1] Intelligent Network Capability Set Number 1 Guidelines and Work Plan Section 2.2 of SWPX/4-1 Meeting Report, CCITT TD439R1, October (1990)

