Packetization and Packet Curves in Network Calculus

Anne Bouillard*
Department of Informatics
Anne.Bouillard@ens.fr

Nadir Farhi
Université Paris-Est, IFSTTAR
nadir.farhi@ifsttar.fr

Bruno Gaujal
INRIA/LIG MESCAL
bruno.gaujal@imag.fr

***Abstract*—Arrival and service curves are core functions in the Network Calculus framework [2], [3]. Based on those curves, we present in this talk a new formalism for data packetization in Network Calculus, the packet curves. Indeed, a more precise knowledge of the packet characteristics can be efficiently exploited to get tighter performance bounds, for example for aggregation of flows, packet-based service policies and shared buffers. Finally, we will give a model for a wormhole switch and show how our results can be used to get efficient delay bounds. Details and proofs can be found in [1].**

a) Packet operator and packet curves: For a server S, A is the arrival flow - $A(t)$ is the cumulative amount of data that arrived until time t - and B is the departure process - $B(t)$ is the cumulative amount of data that left until time t. Suppose that A is made of packets of variable size. One may consider P the *packet flow* associated to this flow, such that $\forall t \geq 0$, $P(t)$ is the number of entire packets that arrive until time t. The transformation of an arrival flow into a packet flow is made using the *packet operator*, which is a function $\mathcal{P} : \mathbb{R} \to \mathbb{N}$ such that for an amount x of arrival data, $\mathcal{P}(x)$ is the number of *entire* packets in x: $P = \mathcal{P} \circ A(t)$.

The operator $\mathcal{P}$ may not be perfectly known, but some information about it may be available, more precise than only the minimum and maximum packet length respectively denoted by $\ell_{\min}$ and $\ell_{\max}$. For example, a flow with packets of size 1 and 2, where in three successive packets, there are at least one packet of size 1 and at least one packet on size 2. In order to take into account this information, we introduce the *packet curve* of a packet operator:

A curve π (resp. Π) is a minimum (resp. maximum) packet curve for $\mathcal{P}$ if $\forall 0 \leq x \leq y$,

$$\mathcal{P}(y) - \mathcal{P}(x) \geq \pi(y-x) \quad (\text{resp. } \mathcal{P}(y) - \mathcal{P}(x) \leq \Pi(y-x)).$$

With the example described above, one can take $\pi : x \mapsto (3/5(x - 2/3))_+$ and $\Pi : x \mapsto 3/4x + 3/2$, where $(x)_+ = \max(0, x)$. Stair-case functions can be more precise, but affine functions and rate-latency functions are easier to handle from an computational viewpoint. Note that if π is defined as $\pi : x \mapsto \mu(x - \nu)_+$, then $\nu \geq l_{\max}$ and $\mu \geq 1/\ell_{\max}$ and if Π is defined as $\Pi : x \mapsto V + Ux$, then $V \leq 1/\ell_{\min}$ and $V \geq 1$.

b) Properties of the packet curves: We assume here that A (resp. A_i, $i \in \{1, 2\}$) is upper-constrained by the arrival curve α (resp. α_i) and has packet operator $\mathcal{P}$ (resp. $\mathcal{P}_i$) with packet curves π and Π (resp. π_i and Π_i) and that β is a (strict) service curve of the server. We also assume a FIFO service per flow. Then, the following properties hold.

(i) Π and π are maximal and minimal packet curves for B.
(ii) $\Pi \circ \alpha$ is an arrival curve for the packet flow P.
(iii) $\pi \circ \beta$ is a minimum (strict) service curve for P.
(iv) If β' be a minimum simple (resp. strict) service curve for a packet flow $P = \mathcal{P} \circ A$, then, $(\Pi)^{-1} \circ \lfloor \beta \rfloor$ (resp. $(\Pi)^{-1} \circ \lceil \beta \rceil$) is a minimum simple (resp. strict) service curve for $(\mathcal{P})^{-1} \circ \mathcal{P} \circ A$.
(v) $\pi_1 * \pi_2$ is a minimum packet curve for the (blind) aggregation of A_1 and A_2.

c) Modeling some network elements: Some network element can be more precisely studied using packet curves. We describe here three examples in this paragraph.

Superposition of periodic flows. When the aggregation of several flow is FIFO, better packet curves than $\pi_1 * \pi_2$ can be found. A special case is the superposition of periodic flow. Consider N flows, where flow n, $1 \leq n \leq N$, is composed of packets of size S_n arriving with period T_n. Set $\rho_n = S_n/T_n$. Then π_N and Π_N are respective minimum and maximum packet curves for the superposition of those flows:

$$\pi(x) = \left(\sum_{n=1}^{N} \frac{x}{T_n \sum_i \rho_i} - \sum_{n=1}^{N} \frac{\sum_i T_i \rho_i}{T_n \sum_i \rho_i} \right)_+$$

$$\Pi(x) = \sum_{n=1}^{N} \frac{x}{T_n \sum_i \rho_i} + \sum_{n=1}^{N} \frac{\sum_i T_i \rho_i}{T_n \sum_i \rho_i}.$$

Note that the rates of the two functions are equal, thus optimal.

Non-preemptive service curves. Figure 1 gives the general scheme of computation, using the basic properties of packet curves. This scheme is more efficient that the computations that may be done using classical methods and taking into account the minimum and maximum packet sizes only when the service policy is based on counting packets. An example of such policy is the Round-Robin. For this policy, three service curves can be computed (we consider two flows, and the residual service curve of the first):

* This author has carried out the work presented in this paper at LINCS (www.lincs.fr).

This work has been funded by the French National Research Agency ANR (PEGASE SEGI 009 02).

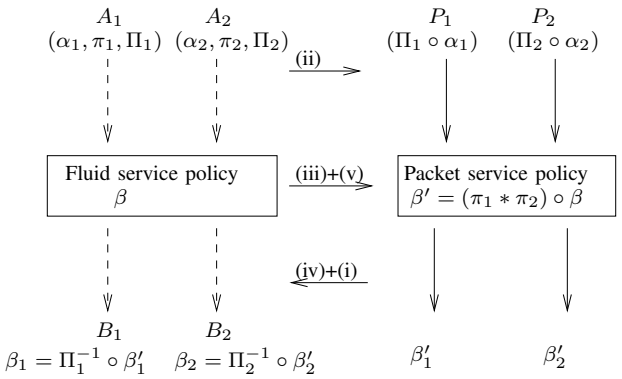

Fig. 1. Non-preemptive service calculus scheme.

- Classical method: $\beta_c(t) = \left(\frac{\ell_i^{\min}}{n\ell^{\max}}\beta - \ell^{\max}\right)^+$,
- Scheme method: $\beta_s(t) = (\Pi_i)^{-1}\left(\frac{1}{n}(\pi_i * \pi_2 \circ \beta) - 1\right)^+$,
- Ad-hoc method: $\beta_{ah}(t) = (Id + \pi_2^{-1} \circ \Pi_1 + 1)^{-1} \circ (\beta - \ell_{\max})_+$.

Numerically, with $\ell_{\min} = 1$, $\ell_{\max} = 2$, $\pi_1 = \pi_2 = \pi$, $\Pi_1 = \Pi_2 = \Pi$ defined above, and $\beta(t) = 10t$, we have $\beta_c(t) = 2.5(t - 0.8)_+$, $\beta_s(t) = 4(t - 0.97)_+$ and $\beta_{ah}(t) = 4.44(t - 0.68)_+$. The residual service rate using the scheme is much better, but the ad-hoc method is the best.

Shared queues. Let A_1 and A_2 be two packetized data flows arriving at the same server. The server serves the packets one by one, and each time it finishes the service of one packet, it picks another packet of either one flow or the other (we still assume FIFO service per flow). Now, the servers provide a different service for those flows: packets of flow A_1 have a strict minimum service curve β_1 and packets of flow A_2 have a strict minimum service curve β_2. When finishing the service of one packet, if the next one is from a different flow, then the service is reinitiated (as if the switching time is the beginning of a backlogged period).

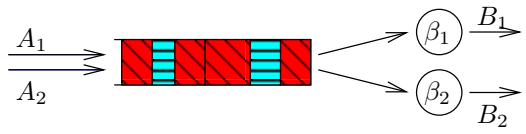

Fig. 2. Shared queues: the two servers cannot be active at the same time.

Considering only the periods of time when server 1 is active (and server 2 idle), $\tilde{\beta}_1$ is a service curve for flow 1 with:

$$\tilde{\beta}_1^{-1}(x) = \left(\beta_1^{(\lfloor \Pi_1(x) \rfloor)}\right)^{-1}(\Pi_1^{-1}(\lfloor \Pi_1(x) \rfloor)) + \beta_1^{-1}(\pi_1^{-1}(0^+)).$$

Then an overall service curve for the shared queue is $\tilde{\beta}_1 * \tilde{\beta}_2$.

This formula is rather pessimistic, but gives the key idea to find better service curves, that is, for each flow compressing time when the server of interest is idle. Improvements are obtained by bounding the number of idle periods and the lengths of those idle periods. This is only possible if minimal and maximal arrival curves are known.

d) Application to performance evaluation in a switch: The talk will be concluded by the presentation of an application of all those network elements to computing delays in a switch with N input ports and N output ports. Input ports serve packets with the FIFO policy (infinite service rate), and output ports use the (packetized) Round-Robin policy (fixed service rate).

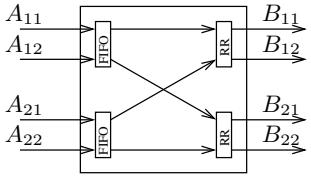

Fig. 3. 2×2 switch.

The steps are the following.

1) Compute individual service curves for the output ports (RR).
2) Compute a service curve for the shared queues.
3) Use the service curves obtained to derive arrival curves for $B_{i,j}$.
4) Iterate using the additional knowledge obtained.

References

[1] A. Bouillard, N. Farhi, and B. Gaujal. Packetization and Aggregate Scheduling. Rapport de recherche RR-7685, INRIA, July 2011.

[2] C. S. Chang. *Performance Guarantees in Communication Networks*. TNCS, 2000.

[3] J.-Y. Le Boudec and P. Thiran. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, volume LNCS 2050. Springer-Verlag, 2001. revised version 4, May 10, 2004.