Worst-Case Communication Time Analysis of Networks-on-Chip with Shared Virtual Channels

Eberle A. Rambo, and Rolf Ernst
Institute of Computer and Network Engineering
TU Braunschweig, Germany
{rambo/ernst}@ida.ing.tu-bs.de

Abstract—Network-on-Chip (NoC) based multi- and many-core architectures show high potential for use in real-time applications considering their superior efficiency. In such systems, it is necessary to provide guarantees that in the worst-case behavior, the application's timing requirements are still met.

I. INTRODUCTION

Network-on-Chip (NoC) based multi- and many-core architectures show high potential for use in real-time applications considering their superior efficiency. In such systems, it is necessary to provide guarantees that in the worst-case behavior, the application’s timing requirements are still met.

Formal real-time analysis methods derive upper and lower bounds for latency and throughput of a network by computing the worst possible interference caused by competing traffic [1]. The worst-case latency bounds for accessing the interconnect are the base to any worst-case response time guarantee and they need to be as tight as possible in order to fully exploit the system performance (over-conservativeness leads to unnecessary low resource usage and even to an apparent unschedulability). An accurate analysis is not trivial and is a known outstanding research problem in NoC design [2].

The majority of research has focused on the analysis of traditional off-chip networks [3]–[5], which can not be directly applied to NoCs [6]. Indeed, in off-chip networks, the topology is not fixed, the behavior of nodes is often not predictable and there are different constraints, such as power and area.

Quality-of-Service (QoS) mechanisms have been widely used to enforce predictability in off- and on-chip networks, simplifying the worst-case analysis and enabling some guarantees. Time-Division Multiple Access (TDMA) has been exploited in [7], [8]. It eliminates interference by separating the transmission time of different streams in time slots. Another option is the use of priorities, investigated in [9]–[11]. By assigning different priorities to each stream, the interference that a stream may suffer is limited to the streams with higher priority. For instance, [11] analyzes the schedulability of a set of streams on a priority-based, wormhole-switched, transmission preemptive NoC, but limits one stream per virtual channel and each virtual channel must have a distinct priority. Although QoS simplifies the worst-case analysis, such mechanisms imply additional hardware overhead and lower resource utilization thus limiting the overall performance. In order to make better use of the shared resources of the network, best-effort networks are preferred.

Best-effort NoCs do not reserve resources but switch packets/flits as soon as possible and the traffic receives equal treatment. This complicates the analysis by introducing direct and indirect interference. These NoCs have been addressed in [1], [12]–[14]. The authors in [13] analyze and provide worst-case latency bounds for packet-switched best-effort NoCs. Bounds for individual traffic streams are given using network calculus. This work was extended in [12] to consider wormhole-switched NoCs. Both works consider only one buffer per input port (i.e. one virtual channel) and a simple round robin scheduler, which is not realistic for a fully-functional real-time multi-core system.

A more efficient and complex two-stage scheduler for input-buffered switches (iSLIP) [15] has been studied in [1], [14]. The work of [14] assumes worst-case traffic injection for all streams, which enables only symmetrical guarantees (i.e. same guarantees for all streams) unless QoS support is added. This limitation was overcome in [1], which models the problem as a response time analysis of multi-core processors with shared resources. The analysis yields tighter bounds and is based on Compositional Performance Analysis (CPA) [16], which enables exploiting the traffic behavior of individual streams and providing non-symmetric guarantees, i.e. a different guarantee to each stream, according to its traffic behavior. Both approaches [1], [14] support more than one virtual channel but assume that every stream uses a different virtual channel.

Real-time multi-core systems will contain more tasks than virtual channels due to the constantly increasing number of tasks mapped to a single system. The tasks will eventually need to share virtual channels. This raises new analysis challenges to be addressed. The authors in [17] present a method for global arbitration of accesses to groups of locally-arbitrated resources (e.g. virtual channels or output ports on an end-to-end path through the NoC). This corresponds to temporal virtual channel reservation which includes reservation setup time, while in our case wormhole routing serializes individual coherent flit streams. This asks for a different analysis. To the best of our knowledge, no worst-case communication time analysis method has been proposed for best-effort, two-stage arbitrated NoCs with shared virtual channels.

The contribution of this paper is a worst-case communi-
cation time analysis of wormhole-switched, best-effort NoCs with two-stage iSLIP arbitration and static routing, extending [1]. No special QoS mechanisms for real-time traffic are assumed and thus all streams are treated equally. The approach supports virtual channel sharing, i.e. more than one stream can use the same virtual channel. The traffic inside a virtual channel follows a FIFO scheduling. The analysis is based on CPA, which allows specifying traffic injection patterns for each stream individually and enables non-symmetric guarantees. We assume that no back-pressure occurs, which can be validated with the worst case analysis provided in this paper.

II. THE NETWORK-ON-CHIP

The analysis assumes a typical NoC for real-time systems implementing wormhole switching, which allows the use of smaller buffers, and deterministic source routing, where both route and virtual channel are defined by the source. Due to wormhole switching, packets consist of fixed-sized flits and the arbitration and buffer space management are performed at a flit granularity.

The routers are input-buffered and there are separate FIFO queues for each virtual channel (VC). The flits are switched using a crossbar switch with one port for each input and output ports, which implies that at most one VC per input port can be switched at a time. The scheduling algorithm implemented by the routers is the 2-stage SLIP. The SLIP algorithm was presented in [15] for conventional networks and has been adapted for NoCs in [18]. The algorithm can be iterative (iSLIP) but the non-iterative variant is considered because only one iteration can be performed in a NoC environment that requires low arbitration latencies, and because it is hard to make safe assumptions on further iterations.

The SLIP scheduler is essentially a two-stage round-robin arbitration scheme. First, each input port sends requests to the output ports being requested by its VCs. The first stage then arbitrates at each output port and grants the request to an input port following a fixed round robin schedule. The first-stage arbitration is also aware of the states of the VCs, since it is a wormhole-switched NoC: The pointer is updated only when the input port accepts the grant, setting the lowest priority to the most recently served. The second stage then arbitrates at the input port, also based on a fixed round robin schedule, selecting one of the VCs that had its request granted in the first stage.

III. COMPOSITIONAL PERFORMANCE ANALYSIS

The analysis is based on Compositional Performance Analysis (CPA). It relies on independent local analyses of the system resources, such as buses and CPUs, and a global analysis loop that assembles the local results [16]. The system model of CPA is based on three components: resources, tasks, event models. Besides providing bounds to worst-case response times and jitter of tasks, CPA also provides bounds to end-to-end latencies and buffer sizes.

The Network-on-Chip is mapped to CPA as a multi-core processor with shared resources, as proposed in [1]. Each output port of the router is modeled as a processing resource and the input port as a shared resource with mutually exclusive access. The mutex-protected shared resource models the limitation of the input port of sending only one flit at a time to an output port. A traffic stream is modeled as a chain of tasks mapped to resources according to its path in the NoC.
minimum number of events that can arrive in a given time interval $\Delta t$. Their pseudo-inverse counterpart functions $\delta^{-}_{\text{in},i}(q)$ and $\delta^{-}_{\text{out},i}(q)$ return the minimum and maximum time interval between the first and last of any sequence of $n$ event arrivals. A conversion method is presented in [19].

The local analysis uses the busy windows approach (former busy period) [20]. It constructs a critical instant, which marks the beginning of the busy window and considers the worst-case arrival sequence of events (input models) and the scheduling policy to build the maximum delay that a task can suffer. This maximizes a task’s response time (time from its activation until the end of its execution). The output event model is then

$$\delta^{-}_{\text{out},i}(q) = \max\{(q - 1) \cdot C, \delta^{-}_{\text{in},i}(q) - J_{i}^{\text{exp}}\} \quad (1)$$

where $(q - 1) \cdot C$ is the best-case execution time, $\delta^{-}_{\text{in},i}(q)$ is the input event model and $J_{i}^{\text{exp}}$ (Section IV-B) is the response time jitter (difference between the worst and best-case execution times).

In the global analysis loop, output event models of tasks are propagated to their dependent tasks, which are activated by the completion of other tasks, and become then their input event models. The analysis stops when all event models are stable (a fix point is reached) or when predefined constraints are violated (e.g. maximum stream latency).

IV. WORST-CASE ANALYSIS

In order to perform a worst-case communication time analysis of the NoC as a performance analysis according to CPA, we need to derive the local analysis. It requires the definition of the $q$-event queueing delay $w_{i}(q)$ (cf. busy-period in [20]). The queueing of a task $\tau_{i}$ is the maximum time from the beginning of the busy-window until $q$-th activation can be transmitted.

To conservatively account for all sources of blocking of a flit’s transmission, we decompose the queueing delay into individual sources of blocking:

- Flit transfer time $C$: the execution time of task $\tau_{i}$ is determined by the execution time excluding any kind of blocking. We consider a flit transfer time equals 1 cycle because the flit has constant size.
- Direct input blocking $B_{i}^{\text{D}}$ and $B_{i}^{\text{D,vc}}$: task $\tau_{i}$ can be blocked by a task $\tau_{j}$, which shares the same input port but not the output port ($\tau_{j} \in \text{In}^{i}$); e.g. task $\tau_{i}$ suffers direct input blocking caused by $\tau_{2}$ and $\tau_{3}$ in Fig. 2.
- Direct output blocking $B_{i}^{\text{DO}}$: task $\tau_{i}$ can be blocked by a task $\tau_{j}$, which shares the output port but not the input port ($\tau_{j} \in \text{Out}^{i}$); e.g. task $\tau_{i}$ suffers direct output blocking caused by $\tau_{2}$ and $\tau_{3}$ in Fig. 2.
- Indirect output blocking $B_{i}^{\text{IO}}$: task $\tau_{i}$ can be indirectly blocked by a task $\tau_{j}$, which shares the same output port but not the input port ($\tau_{j} \in \text{Out}^{i}$), because of another task $\tau_{k}$, which shares the input port with $\tau_{j}$ ($\tau_{k} \in \text{In}^{i}$). E.g. task $\tau_{3}$ experiences indirect output blocking through $\tau_{2}$ and $\tau_{3}$ caused by $\tau_{i}$ in Fig. 2.
- Overlap blocking $B_{i}^{\text{OV}}$ and $B_{i}^{\text{OV,vc}}$: task $\tau_{i}$ can be blocked by a task $\tau_{j}$, which shares the same input and same output port ($\tau_{j} \in \text{Overlap}^{i}$), i.e. an overlapping stream. E.g. tasks $\tau_{2}$ and $\tau_{3}$ block each other in Fig. 2.

In a router implementing the SLIP arbitration scheme, the queueing delay $w_{i}(q, q_{i}^{a})$ is then given by the sum of all the individual sources of blocking:

$$w_{i}(q, q_{i}^{a}) = (q - 1) \cdot C + B_{i}^{\text{D,vc}}(q_{i}^{a}) + B_{i}^{\text{OV,vc}}(q_{i}^{a}) + B_{i}^{\text{DO}}(w_{i}(q, q_{i}^{a}), q) + B_{i}^{\text{IO}}(w_{i}(q, q_{i}^{a}), q) + B_{i}^{\text{D,vc}}(w_{i}(q, q_{i}^{a})) + B_{i}^{\text{D,vc}}(w_{i}(q, q_{i}^{a}))$$

where $C$ is the flit transfer time in cycles and $q_{i}^{a}$ is the arrival time of the $q$-th activation of $\tau_{i}$.

Typically seen in busy-window based scheduling analyses, Eq. 2 forms an integer fixed point problem, as $w_{i}(q, q_{i}^{a})$ appears on both sides of the equation. It can be solved iteratively, starting with $w_{i}(q, q_{i}^{a}) = (q - 1) \cdot C$.

A. Sources of Blocking

Now we derive the individual sources of blocking. It requires defining two auxiliary functions $\rho_{i}^{+}(\Delta t)$ and $\rho_{i}^{-}(q)$.

Let $\rho_{i}^{+}(\Delta t)$ be the maximum number of flits that can arrive in a time interval $\Delta t$ at a stream $i$ considering whole packets.

$$\rho_{i}^{+}(\Delta t) = \lceil \eta_{i}^{q}(\Delta t) / \text{size}(i) \rceil \cdot \text{size}(i) \quad (3)$$

where $\text{size}(i)$ is the size of packet of stream $i$ in flits.

Let $\rho_{i}^{-}(q)$ be the minimum distance between arrivals of the first flit and the head flit of the packet containing the $q$-th flit at a stream $i$.

$$\rho_{i}^{-}(q) = \delta^{-}_{i}(\lceil (q / \text{size}(i)) - 1 \rceil \cdot \text{size}(i) + 1) \quad (4)$$

The direct output blocking $B_{i}^{\text{DO}}(\Delta t)$ can be bounded by

$$B_{i}^{\text{DO}}(\Delta t, q) = \sum_{c \in \text{V}(i)} C \cdot \min\left\{ q, \sum_{j \in \text{Out}^{i}} \eta_{j}^{q}(\Delta t) \right\} + \sum_{j \in \text{Out}^{i}(c)} \rho_{j}^{+}(\Delta t) \cdot C \quad (5)$$

where $\text{V}(i)$ is the set of all VCs except the virtual channel used by stream $i$; $\text{V}(i)$ is the virtual channel used by stream $i$; $\text{Out}^{i}_{c}$ is the set of streams mapped to the same output port and to virtual channel $c$; $\text{Out}^{i}(c)$ is the set of streams mapped to the same output port and to the same virtual channel as stream $i$.

The first term of Eq. 5 accounts for the blocking caused by streams using different VCs. Due to round robin, stream $i$ may be blocked at most $q$ times for $q$ activations of stream $i$ and also no more than the sum of activations of all tasks sharing a given VC, since accesses to a same VC are serialized. That is captured by the min-function. The second term of Eq. 5 accounts for the blocking caused by streams using the same VCs. Due to wormhole switching, once the scheduler at an output port grants access to a VC to an input port, no other input port can access the same VC and output port until the corresponding VC-reservation is released, i.e. the packet is fully transmitted. This is captured by $\rho_{j}^{+}$, which considers that, when a head flit from $j$ arrives within the time interval $\Delta t$, the whole packet will be served before $i$.

The indirect output blocking $B_{i}^{\text{IO}}(\Delta t)$ is bounded by

$$B_{i}^{\text{IO}}(\Delta t) = \sum_{c \in \text{V}(i)} \min\left\{ q \cdot C, \sum_{j \in \text{Out}^{i}(c)} B_{j}^{\text{D,vc}}(\Delta t, \eta_{j}^{q}(\Delta t)) \right\} + \sum_{j \in \text{Out}^{i}(c)} B_{j}^{\text{D,vc}}(\Delta t, \rho_{j}^{+}(\Delta t)) \quad (6)$$
This kind of blocking happens when a stream \( j \) which shares the output port with stream \( i \), wins the arbitration at the output port but loses the arbitration at the input port to another stream \( k \). The first term of Eq. 6 accounts for the direct input blocking of \( j \) when \( v(j) \neq v(i) \). Due to round robin on the output port, stream \( i \) may be blocked at most \( q \) times for \( q \) activations. This is captured by the min-function. The second term accounts for the direct input blocking of \( j \) when \( v(j) = v(i) \). Due to wormhole switching, we have to consider the blocking suffered by the activations of entire packets of \( j \). The cases when \( v(j) = v(k) \) have no impact on the worst case, since considering any activation of \( k \) before \( j \) would only maintain or decrease the indirect blocking experienced by \( i \).

The blocking caused by streams on other VCs \( B_{i \text{vc}}^{\text{DJ}}(\Delta t) \) is bounded by

\[
B_{i \text{vc}}^{\text{DJ}}(\Delta t) = \sum_{c \in V(i)} \sum_{j \in \text{Overlap}^j_{v(i)}} \eta_j^+(\Delta t) \cdot C
\]

where \( \text{Overlap}^j_{v(i)} \) is the set of streams mapped to the VC \( c \) and to the same input and output ports as stream \( i \).

Eq. 7 sums the direct interference from another overlapping stream \( j \). Since streams \( i \) and \( j \) share the same input port and output port, they share exactly the same interferers. And because of that, all interference experienced by \( j \) has already been accounted for when analyzing the interference for \( i \) before considering overlapping streams. Moreover, the round robin scheduling does not help in the worst case, because it is possible that streams \( i \) and \( j \) “steal” the output port grant of each other, therefore we must conservatively assume that every activation of \( j \) will be served before \( i \) [1].

The blocking caused by overlapping streams sharing a VC \( B_{i \text{vc}}^{\text{OV}}(a_i^q) \) differs from the other blocking sources because of the FIFO scheduling inherent in a virtual channel. The worst-case response-time analysis under FIFO scheduling has been studied in [21]. Thus, an upper bound for the blocking is

\[
B_{i \text{vc}}^{\text{OV}}(a_i^q) = \sum_{j \in \text{Overlap}^j_{v(i)}} \rho_j^+(a_i^q) \cdot C
\]

where \( a_i^q \) is the arrival time of the \( q \)-th event of stream \( i \).

In this case, the interference is caused by complete packets, since there is no flit-interleaving between different packets in the same virtual channel. This is captured by \( \rho_j^+ \). Moreover, the packets follow a FIFO scheduling at the input port. Thus, Eq. 8 sums the largest number of packets of \( j \) that may arrive before and which will therefore be served before the \( q \)-th activation of \( i \) [21].

The direct input blocking is also differentiated between blocking caused by streams on other VCs \( B_{i \text{vc}}^{\text{DI}}(\Delta t) \) and the blocking caused by streams on the same VC \( B_{i \text{vc}}^{\text{DJvc}}(a_i^q) \).

The direct input blocking caused by streams using different virtual channels \( B_{i \text{vc}}^{\text{DI}}(\Delta t) \) can be bounded by

\[
B_{i \text{vc}}^{\text{DI}}(\Delta t, q) = \sum_{c \in V(i)} C \cdot \min \left\{ q, \sum_{j \in I_i} \eta_j^+(\Delta t) \right\}
\]

where \( I_i \) is the set of streams mapped to VC \( c \) and to the same input port as stream \( i \) and \( \eta_j^+(\Delta t) = \eta_j^+(\Delta t + R_i^q) \) is the the maximum number of requests that may be issued by a stream \( i \) for an input port in a given time interval \( \Delta t \) [1].

Due to round robin at the input port, stream \( i \) may be blocked at most \( q \) times per each VC in use, but also no more than the number of task activations seen on that VC.

The direct input blocking \( B_{i \text{vc}}^{\text{DJvc}}(\Delta t) \) caused by streams using the same virtual channel needs not only to consider the transmission time of stream \( j \)’s flits but also the direct and indirect output blocking that they experience. The blocking \( B_{i \text{vc}}^{\text{DJvc}}(\Delta t) \) is bounded by

\[
B_{i \text{vc}}^{\text{DJvc}}(a_i^q) = \sum_{j \in I_i} \left\{ \rho_j^+(a_i^q) \cdot C \right\} + B_{i \text{vc}}^{\text{DO}}(\delta_j^+(a_i^q)) - C
\]

The blocking caused by other streams in the same virtual channel at the input consists on the number of packets of \( j \) that will be transmitted before \( i \) and the blocking that those packets will suffer. Since \( j \) uses the same input port as \( i \) and since its activations are included in the busy-window of \( i \), the interference at the input port (\( B_{i \text{vc}}^{\text{DI}} \) and \( B_{i \text{vc}}^{\text{DJvc}} \) is already being considered. The interference seen by \( j \) on the output port however needs to be included. Therefore, Eq. 10 sums the time needed to execute \( \rho_j^+(a_i^q) \) activations and sums the direct and indirect output blocking experienced until the completion of those activations.

B. Metrics for a single router

After deriving the worst-case queueing time for \( q \) activations in a router, we can bound the worst-case response time \( R_i(q) \) for the \( q \)-th activation of stream \( i \) by finding the maximum distance between the processing of \( q \) activations \( (w_i(q, a_i^q)) \) where \( q \) is the arrival of the \( q \)-th activation \( a_i^q \).

\[
R_i(q) = \max_{a_i^q \in A_i^q} \{ w_i(q, a_i^q) - a_i^q \} + C + O_r
\]

where \( O_r \) is the routing overhead at the router.

The equation forms a nested optimization problem: for every number of activations \( q \) within a busy-time, all possible arrival times \( a_i^q \) of the \( q \)-th event have to be considered to find the maximum response time [21]. In the NoC, the number of these arrival candidates can be reduced to a finite amount by exploiting the fact that time can be discretized to clock cycles. It has been shown in [21] that it is sufficient to consider only arrival candidates \( a_i^q \) that coincide with activations of the interfering workload \( I(i) \). Also, considering an earlier or later activation of \( i \) would only reduce the interference.

\[
A_i^q = \bigcup_{j \in I(i)} \{ \delta_j^-(n) \mid \delta_j^+(n) \leq \delta_j^-(n) < \delta_j^+(n + \text{size}(i)) \}_{n \geq 1}
\]

where \( I(i) = \text{In}_{v(i)}^i \cup \text{Overlap}^j_{v(i)} \) is the considered interfering workload.

The worst-case response time or single hop latency for a flit of stream \( i \) at a router is the maximum among all \( R_i(q) \).

\[
R_i^q = \max_{1 \leq q \leq \hat{q}_i} \{ R_i(q) \}
\]

where \( \hat{q}_i = \eta_i^+(\hat{w}_i) \) is the maximum number of activations in the longest busy window \( \hat{w}_i \).
The response time jitter \( J^{r_{exp}}_i \) of a flit of stream \( i \) in a router is derived from the difference between it’s worst- and best-case response times.

\[
J^{r_{exp}}_i = R^+_i - R^-_i
\]  

where a conservative lower bound for \( R^-_i \) is \( C + O_p \).

The worst-case service \( \beta_i^- (\Delta t) \) is the minimum number of served flits guaranteed for stream \( i \) in a time interval \( \Delta t \) [1].

\[
\beta_i^- (\Delta t) = \max_{\forall q \geq 0} \{ q \mid R_i(q) \leq \Delta t \wedge q < \eta_i^-(R_i(q)) \}
\]  

The worst-case backlog of a stream \( i \) \( b_i \) is the maximum number of unserved flits and can be calculated as follows [1]:

\[
b_i = \max\{\eta_i^+(\Delta t) - \beta_i^- (\Delta t)\}
\]  

The analysis assumes that back-pressure does not occur. This means that \( b_i \) does not exceed the available buffer capacity. It can be validated during analysis.

C. Metrics for the network
After the analysis reaches a stable point (i.e. it is schedulable), we can calculate the metrics for the whole network.

The maximum end-to-end latency \( l_i^+(\Delta t) \) for \( q \) activations of stream \( i \) can be bounded by the time it takes to inject \( q \) flits in the network, the worst-case response time for each hop in its path, and overheads.

\[
l_i^+(\Delta t) = \delta_{First(i)}^-(q) + O_p + \sum_{j \in Tasks(i)} R^+_j
\]  

where First\((i)\) denotes the first task, Tasks\((i)\) is the set of all tasks of stream \( i \), and \( O_p \) is the de/packetization overhead.

The maximum injection throughput \( r_i^+(\Delta t) \) is the number of bytes that can be injected by stream \( i \) within the time interval \( \Delta t \) [1]. It is given by

\[
r_i^+(\Delta t) = f \cdot \eta_{First(i)}^- (\Delta t) \div \Delta t
\]  

where \( f \) is the size of a flit (bytes).

V. EXPERIMENTS
The proposed analysis was evaluated and compared with simulation and two other analyses: the baseline approach [1] ("baseline") and a version of [14] ("symmetric") extended to support overlapping streams. The analyses were implemented in the PyCPA framework [19]. The simulations were carried out using the OMNeT++ network simulation framework [22] and HNOCs library [23].

The system analyzed in the experiments is the one presented in the example (Figures 1 and 2), containing four streams. The system configuration is relatively simple but shows the relevant aspects of the analysis. The streams injected packets periodically in the NoC according to the following input event models: stream 1 has 2 activations (i.e. a 2-flit packet) every 9 cycles; stream 2 has 5 activations every 50 cycles (except in Figure 3, where they vary); stream 3 has 2 activations every 30 cycles; and stream 4 has 3 activations every 21 cycles. The virtual channel configuration varies according to the experiment. A 4-cycle routing overhead at each router \((O_r)\) and a 10-cycle de/packetization \((O_p)\) overhead were also considered.

Since the CPA consists of local analyses and the propagation of their results, we first show and compare the results of the worst-case response time of the tasks (WCRT) in a single router. Then we address the results for the whole network.
We provided experimental evaluation and validation against simulation, which showed that the values are conservative and tighter than previous work. Techniques, such as high level arbitration, can profit from the tight bounds from this analysis in order to provide their own worst-case bounds.

REFERENCES


VI. CONCLUSION

In this paper we have presented a worst-case communication time analysis of best-effort NoCs with a two-stage SLIP arbitration. The analysis provides worst-case latency bounds for streams individually and is able to do so also for streams that share the same virtual channel and the same path.