Efficient Traffic Splitting in Parallel TCP-Based Wireless Networks: Modelling and Experimental Evaluation

G.J. Hoekstra[1,2], R.D. van der Mei[2,3] and J.W. Bosman[2]

[1]Innovation Research & Technology, Thales Nederland B.V., Huizen, The Netherlands
[2]CWI, Department of Stochastics, Amsterdam, The Netherlands
[3]VU University Amsterdam, Department of Mathematics, The Netherlands

***Abstract*—The concurrent use of networks provides a powerful means to boost performance in areas covered by multiple networks where only limited bandwidth is available. However, despite its enormous potential for performance improvement only little is known about how to effectively exploit the potential for performance improvement in practical deployments. This raises the need for traffic-splitting-and-reassembly algorithms that are effective, yet simple and easy-to-deploy. Motivated by this, we first propose a simple analytic flow-level model, called the Concurrent Access Network (CAN) model, that optimally splits traffic in the idealized situation where there is full state information at infinitely fine-grained time granularity, leading to zero synchronization delay during the reassembly phase. Next, we present a new splitting algorithm for TCP-based networks that uses a simple score function to make on-the-fly decisions on the routing of individual TCP segments, based on the measured per-connection RTT, transmission-buffer content and throughput. Then, we use the CAN-model as a benchmark to evaluate the effectiveness and practical usefulness of the score-function based algorithm on real TCP networks in a test-lab environment. Extensive lab experimentation demonstrates that this score-function based splitting of TCP traffic is extremely efficient, leads to close-to-optimal response-time performance and is easily deployable.**

Keywords*—*Traffic Splitting, Processor Sharing, Concurrent Access, Flow-level Performance, File Splitting.

I. Introduction

Users in areas covered by a multitude of access networks may benefit from the opportunity to combine the available network resources to improve performance. This potential benefit is particularly appealing in the context of wireless networks, where the existing spectral efficiencies leave room for only minor improvements in channel capacity [1]. Therefore, the concurrent use of multiple networks is a viable alternative to better utilize a wider range of frequency bands [2] and to improve on application performance. Concurrent Access (CA) is not a new field of research, but a rather dispersed one, primarily into the areas of queueing theory and communication-protocol implementations. The literature leaves a clear gap between theory and practice: The existing theory (references are listed below) provides important fundamental insight, but does not explicitly tell how the application performance can be boosted by state-of-the-art protocol implementations on CA for transferring files over multiple networks in practical situations. Motivated by this, the aim of this paper is to make a first step towards filling this gap by proposing and evaluating a new, efficient, yet easy-to-deploy algorithm for (near-)optimal TCP-level splitting of traffic in areas covered by a multitude of networks.

The use of multiple fixed-line Public Switched Digital Network (PSDN) channels in parallel can be viewed as an early (1994) realization of CA [3]. Many different forms of parallelism occur throughout different protocol layers in communication systems to enhance reliability, e.g. protecting working channels that transfer voice signalling using SCTP, or to increase network performance for Wireless LANs (WLAN) using multiple antennas in its most recent 802.11n standard [4]. Many research efforts that are focused on combining the capacity of multiple networks concentrate on the link layer, the transport layer, and on the application layer of communication systems. At the *data link layer*, approaches have been proposed for switching between several homogeneous networks [5] and scheduling over heterogeneous networks [6], [7]. However, approaches at this layer require modifications for each different network interface that needs to be supported and, moreover, switching the data segments of the same TCP session over different network links, even in homogeneous networks, adversely affects TCP performance [5], [8]. At the *transport layer*, two main areas can be distinguished: one concentrating on the use of SCTP (see [9], [10], [11]) and the other on using or modifying TCP. Within the IETF the SCTP transport protocol [12] has evolved from a transport protocol for voice-signalling traffic into one that allows various types of information to be transported over different network paths. It needs to be pointed out that the functionality for efficiently using concurrent paths is not considered by the standard, meaning that distributing and re-sequencing the data should be implemented separately, and that the flow- and congestion control mechanism is the same for the possibly different networks used in parallel, which is not in the interest of overall efficient link utilization nor application performance [8], [13]. Several proposals to modify, use and extend TCP for exploiting multiple networks have appeared. The most prominent being a modification of TCP, called pTCP [8], later

followed by mTCP [13] that both aimed at enhancing TCP to be capable of using multiple networks concurrently. Finally, at the *application layer*, another approach can be applied to establish multiple connections to one or more endpoints for transferring information in a distributed manner, as is done by P2P techniques.
In this paper we concentrate on the transport layer to realize CA, because it preserves existing applications, abstracts from network particularities and seems the most promising layer to operate on with respect to achieving high performance gains.

On the theoretical side, there is a wealth of literature on performance models and analysis. Processor Sharing (PS) models have been successfully applied to model TCP-based file transfers over a wide range of different network standards that are commonly used, from cellular [14], [15] to WLANs [16], [17]. The complex dynamics of multiple protocol layers can be modelled to obtain very accurate download response time predictions over a single network [17]. Despite the applicability of PS-based models to real communication networks, there is little known on PS-based models suitable for modelling the use of multiple networks concurrently. As an exception, Key et al. investigate the efficiency of combining multipath routing and congestion control in TCP-based networks. In [18] they show that under certain conditions the allocation of flows to paths is optimal and independent to the flow control algorithm used. In [19] it is shown that with RTT bias uncoordinated control can lead to inefficient equilibria, while without RTT bias, both coordinated and uncoordinated Nash equilibria correspond to desirable welfare maximizing states. The distribution and re-assembly of tasks are typically modelled by fork-join constructions [20], in many cases embedded in so-called stochastic activity networks. In cases where the processing times of the subtasks are independent, exact or numerical analysis is relatively simple (e.g., [21]), whereas the inclusion of dependent processing times (e.g., due to queuing or job splitting) typically leads to very complex analysis (e.g., [22], [23]) and no closed-form solution exists. For PS-based nodes that process the tasks of a job in parallel, the complex correlation structure between the sojourn times at the PS nodes makes an exact detailed mathematical analysis of the model impossible. In [24] and [25], the author analyzes a similar model but with FCFS queues and with probabilistic splitting. We further refer to Altman et al. [26], who consider routing policies in a distributed versus centralized environment. In general our queueing model falls within the framework of fork-join queueing networks, see [27] for an extensive overview. In a recent paper [28], the theoretical foundation for a tail-optimal splitting rule is provided for light foreground load that is shown to work well with respect to both the tail asymptotics and the mean sojourn times.

Despite the fact that the literature on analytic models provides important and valuable insight in the performance implications and efficiency of traffic-splitting algorithms, they do not explicitly reveal how to exploit contemporary protocol implementations on CA in practical deployments. Motivated by this, the aim of this paper is to propose and evaluate a simple, yet efficient and easily deployable traffic-splitting algorithm for TCP-based networks. To this end, we first propose a model that optimally splits traffic in a dynamic way based on full state information at infinitely fine time granularity. This model is called the concurrent access network (CAN) model and will be used as a benchmark. We show that the expected response time under this optimal splitting performance can be numerically calculated by solving a continuous-time Markov chain. In practical deployments, however, there is only limited and coarse-grained information (e.g., measured RTTs, queue lengths and throughputs) is available to base routing decisions on, so that this optimal performance - which will be used as a benchmark - can not be reached. Next, motivated by the work in [29], we propose a simple and easily deployable method for TCP-level traffic splitting, where TCP segments are routed based on a score-function that dynamically adapts the routing decisions to observed per-node RTTs, transmission-buffer contents and measured throughput values. Then, we present the results of extensive test-lab experiments to assess the effectiveness of our approach. The results show that the score-function method matches the performance of the benchmark model extremely well for a wide range of parameter settings, and as such provides a powerful means to effectively split TCP traffic in the presence of concurrently available access networks.

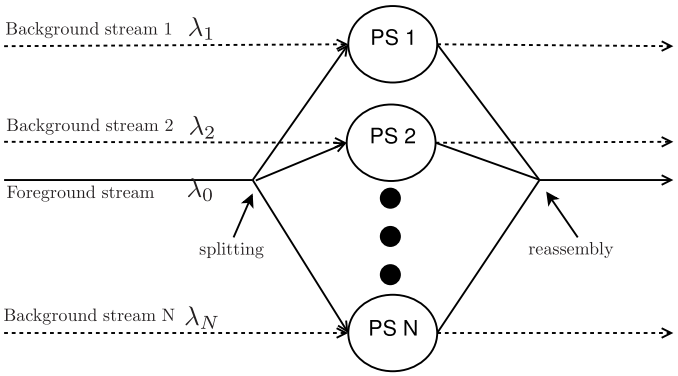

Fig. 1: The concurrent access network model.

The remainder of the paper is organized as follows. In Section II we describe the CAN model and introduce the notation. In Section III we assess the performance of the CAN-model by simulations, and show that the expected response time under the CAN-model is nearly insensitive to the job-size distribution. This allows us approximate the optimal response-time performance by solving the steady-state distribution of a continuous-time Markov chain. In Section IV we discuss a simple score-function based approach to dynamically split traffic at the TCP-layer over different parallel networks. In Section V we discuss the results of extensive experimentation in a test-lab environment. Finally, in Section VII we address a number of topics for further research.

II. The Concurrent Access Network Model

The analytic model consists of N parallel PS nodes (see Figure 1). There are $N+1$ traffic streams: a single stream of foreground jobs (called class-0 jobs) and N streams of background jobs (called class-i jobs, for $i = 1, \ldots, N$). Class-i jobs arrive according to independent Poisson processes

with rates λ_i, the service times are generally distributed with mean β_i, and the corresponding load offered to the system is $\rho_i = \lambda_i \beta_i$, $i = 0, 1, \ldots, N$. Foreground jobs use the capacity of all N nodes simultaneously in a fluid-like manner, using the (instantaneously) available capacity at all the N PS nodes; at any moment in time the capacity available at node i is equally shared amongst all foreground jobs in the system and with the background jobs at node i. The splitting operates without delay with infinitely small granularity and has perfect information about the number of foreground and per-class background jobs in the system. If upon arrival of a tagged foreground job F there are k_0 other foreground jobs in the system and k_i background jobs at node i, then F obtains a fraction

$$f_i := \frac{1}{k_0 + 1 + k_i} \tag{1}$$

of the capacity of node i, for $i = 1, \ldots, N$. Note that in this way, the instantaneous total transmission speed that F receives equals $\sum_{i=1}^{N} f_i$, and that this speed changes during the course of the sojourn time of F in the system, as other jobs may come and go.

For the model under consideration, let S_i be the sojourn time of an arbitrary class-i job in the system. In this paper, our focus is on their expected values, $\mathbb{E}[S_i] \;\; (i = 0, 1, \ldots, N)$.

Remark 1 (Optimality of the CAN model): The CAN-model described above uses fluid-like splitting of foreground jobs at infinitely fine time granularity. Therefore, the performance of the CAN-model is optimal in the sense that the *synchronization delay* in the reassembly phase (which is usually encountered when splitting is done at coarse-grained granularity or with non-perfect or delayed information) is *zero*, while the foreground jobs receive no more than their fair share of capacity at each of the nodes. It is evident that from the viewpoint of the foreground traffic even better performance for foreground jobs can be obtained by allowing unfair capacity sharing at the PS nodes in favor of foreground traffic (see also Section 6).

Remark 2 (Parameterization of TCP-based networks into PS-models): The CAN-model is a flow-level performance model that describes the elastic behavior of TCP-based networks by PS-models, abstracting from the complex packet-level dynamics at the TCP layer. To model the flow-level behavior of TCP-based networks, the parameters of TCP networks (e.g., RTTs, maximum window sizes, maximum segmant sizes, etc.) need to be parameterized into the parameters of the PS-models (e.g., service times). To this end, in Hoekstra and Van der Mei [17] we propose the concept of *effective service time* and give a full parameterization, translating the TCP-parameters into PS-model parameters. In doing so, the transfer time of a file over a TCP network is modeled by the sojourn time of a job in the corresponding PS-model.

Remark 3 (Case of exponential job-size distributions): In the special case where the service-time distributions are all exponentially distributed, with rates $\mu_i := 1/\beta_i$ $(i = 0, 1, \ldots, N)$, then the evolution of the system can be described as a continuous-time Markov chain (CTMC) with state space $S = {\mathbb{N}_0}^{N+1}$, where each state is of the form $s = (k_0, k_1, \ldots, k_N) \in S$, with k_0 the number of foreground jobs in all N PS nodes and k_i $(i = 1, \ldots, N)$ the number of background jobs in PS node i. For each arriving foreground job, a task is assigned to each PS node of which the processing demand will be adjusted to complete the service of all tasks corresponding to the same job simultaneously. It is readily verified that the state-transition rates of the CTMC are as follows:

$$q(s, s + e_i) = \lambda_i \quad (i = 0, 1, \ldots, N), \tag{2}$$

$$q(s, s - e_0) = \sum_{i=1}^{N} \frac{k_0}{k_0 + k_i} \mu_0, \tag{3}$$

$$q(s, s - e_i) = \frac{k_i}{k_0 + k_i} \mu_i \quad (i = 1, \ldots, N), \tag{4}$$

for all possible state combinations in S; here, e_i stands for the unit vector that has zeros on all dimensions except the dimension that corresponds to the total number of foreground jobs (by taking $i = 0$) or number of background jobs (for $i = 1, \ldots, N$) respectively. Equation (2) represents the external arrivals of class-i jobs, for $i = 0, 1, \ldots, N$. Equations (3) and (4) represent the departure of a foreground job, and a class-i job, respectively. Given the stationary distribution of the CTMC, $\pi(\cdot)$, the expected number of foreground and background jobs in the system is, for $k = 0, 1, \ldots, N$,

$$\mathbb{E}\left[N_k\right] = \sum_{i_0=0}^{\infty} \sum_{i_1=0}^{\infty} \cdots \sum_{i_N=0}^{\infty} i_k \pi(i_0, i_1, \ldots, i_N). \tag{5}$$

Using Little's formula, we obtain the expected sojourn time of the foreground and background jobs:

$$\mathbb{E}\left[S_k\right] = \frac{\mathbb{E}\left[N_k\right]}{\lambda_k} \quad (k = 0, 1, \ldots, N). \tag{6}$$

III. Performance of the CAN-model and near-insensitivity

In general, the job-size distributions are not exponential and the analytic model does not allow for exact analysis (see also Remark 3 above). Therefore, we have used simulations to assess the performance under the CAN-model in terms of the mean sojourn times of the foreground traffic streams (i.e., $\mathbb{E}[S_0]$) and the background traffic (i.e., $\mathbb{E}[S_i],\ i = 1, \ldots, N$). To this end, we have performed extensive simulations to obtain the expected response times for a wide variety of scenarios. The job-size distributions were varied as deterministic (with squared coefficient of variation $c_B^2 = 0$), exponential ($c_B^2 = 1$), two-phase hyperexponential (with balanced means and squared coefficient of variation $c_B^2 = 16$), Pareto-3 (i.e. with $Pr\{B > x\} = (1 + x/2)^{-3}$, $x \geq 0$, and hence $c_B^2 = 3$) and Pareto-2 (i.e., with $Pr\{B > x\} = \frac{1}{4x^2}$, $x \geq \frac{1}{2}$, and hence $c_B^2 = \infty$), all with mean 1. To limit durations of the simulation runs, we considered scenarios with $N = 2$ networks, which is the most common CA-scenario in practical deployments. Also, to limit the number of scenarios, in all cases considered the squared coefficients of variation of the job-size distributions were taken to be the same for all classes (i.e., $c_{B_i}^2 = c_B^2$ for $i = 0, 1, \ldots, N$). The foreground load ρ_0 was varied as 0.1, 0.9 and 1.8 to cover load values ranging from light to heavy foreground load. The background-load combinations (ρ_1, ρ_2) were varied such that ρ_1 and ρ_2 take

TABLE I: $\mathbb{E}[S_0]$ under lightly loaded foreground traffic ($\rho_0 = 0.1$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.1	0.3	0.5	0.7	0.9
deterministic	0	0.1	0.57				
exponential	1	0.1	0.57				
Pareto-3	3	0.1	0.57				
hyper-exponential	16	0.1	0.57				
Pareto-2	∞	0.1	0.57				
deterministic	0	0.3	0.63	0.70			
exponential	0	0.3	0.63	0.70			
Pareto-3	3	0.3	0.63	0.70			
hyper-exponential	0	0.3	0.63	0.70			
Pareto-2	∞	0.3	0.63	0.70			
deterministic	0	0.5	0.70	0.80	0.94		
exponential	1	0.5	0.70	0.81	0.95		
Pareto-3	3	0.5	0.71	0.81	0.95		
hyper-exponential	16	0.5	0.71	0.82	0.96		
Pareto-2	∞	0.5	0.71	0.81	0.96		
deterministic	0	0.7	0.80	0.95	1.16	1.53	
exponential	1	0.7	0.81	0.95	1.17	1.55	
Pareto-3	3	0.7	0.81	0.96	1.18	1.57	
hyper-exponential	16	0.7	0.82	0.97	1.20	1.59	
Pareto-2	∞	0.7	0.82	0.97	1.19	1.58	
deterministic	0	0.9	0.99	1.23	1.64	2.51	6.51
exponential	1	0.9	1.00	1.24	1.65	2.53	6.55
Pareto-3	3	0.9	1.00	1.24	1.66	2.55	6.60
hyper-exponential	16	0.9	1.00	1.25	1.67	2.57	6.63
Pareto-2	∞	0.9	1.00	1.25	1.67	2.57	6.67

values 0.1, 0.3, 0.5, 0.7 and 0.9 (of course, only for the parameter combinations for which the system is stable), hence covering scenarios with both light- and heavy-traffic and varying degrees of asymmetry. The results of our experiments are outlined in Tables I, II, and III.

All simulation runs are based on at least 10^8 (and 10^9 for very high load values) foreground observations, leading to accurate predictions of $\mathbb{E}[S_0]$, such that all digits shown below are significant; confidence intervals have been omitted for compactness of the presentation. Note that as an alternative to the simulations, $\mathbb{E}[S_0]$ can also be calculatied numerically for the special case of exponential job-size distributions by using Equations (1)-(6); truncation of the state space was done at a sufficiently large size in order no to influence the results. Table I shows the results for lightly loaded foreground traffic, with $\rho_0 = 0.1$. To avoid duplication due to symmetry, the results are shown for $\rho_1 \leq \rho_2$. The results in Table I reveal two interesting observations. First, we see that in all cases the expected foreground sojourn time $\mathbb{E}[S_0]$ is nearly insensitive to the job-size distribution, and only slightly increases for larger values of c_B^2. We reemphasize that the results in Table I cover a wide variety of scenario's, ranging from light to heavy background loads, from symmetric to strongly asymmetry load values, and from deterministic to highly variable job-size distributions. In addition, we observe that the "exact" values based on (1)-(6) match the simulation results very closely, as it should. Tables II and III below show the results for medium-loaded (with $\rho_0 = 0.9$) and heavily-loaded foreground traffic (with $\rho_0 = 1.8$), respectively. They confirm that the results for the light foreground traffic in Table I are also valid for medium- and heavily-loaded foreground traffic. Again, we observe the $\mathbb{E}[S_0]$ is nearly insensitive to the job-size distribution, and that the CAN-model predictions from (1)-(6) are highly accurate.

TABLE II: $\mathbb{E}[S_0]$ under medium-loaded foreground traffic ($\rho_0 = 0.9$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.1	0.3	0.5
deterministic	0	0.1	1.06		
exponential	1	0.1	1.07		
Pareto-3	3	0.1	1.07		
hyper-exponential	16	0.1	1.07		
Pareto-2	∞	0.1	1.07		
deterministic	0	0.3	1.29	1.69	
exponential	1	0.3	1.30	1.72	
Pareto-3	3	0.3	1.31	1.74	
hyper-exponential	16	0.3	1.32	1.76	
Pareto-2	∞	0.3	1.32	1.76	
deterministic	0	0.5	1.67	2.59	6.98
exponential	1	0.5	1.69	2.63	7.06
Pareto-3	3	0.5	1.72	2.68	7.18
hyper-exponential	16	0.5	1.74	2.71	7.15
Pareto-2	∞	0.5	1.74	2.73	7.45
deterministic	0	0.7	2.52	6.46	
exponential	1	0.7	2.56	6.55	
Pareto-3	3	0.7	2.60	6.71	
hyper-exponential	16	0.7	2.63	6.70	
Pareto-2	∞	0.7	2.66	7.05	
deterministic	0	0.9	6.71		
exponential	1	0.9	6.76		
Pareto-3	3	0.9	6.84		
hyper-exponential	16	0.9	6.83		
Pareto-2	∞	0.9	7.04		

TABLE III: $\mathbb{E}[S_0]$ under heavy-loaded foreground traffic ($\rho_0 = 1.8$).

job-size distribution	c_B^2	ρ_2 \ ρ_1	0.00	0.05
deterministic	0	0.05	6.52	9.55
exponential	1	0.05	6.52	9.57
Pareto-3	3	0.05	6.54	9.60
hyper-exponential	16	0.05	6.54	9.60
Pareto-2	∞	0.05	6.55	9.65
deterministic	0	0.10	9.51	18.62
exponential	1	0.10	9.53	18.64
Pareto-3	3	0.10	9.57	18.75
hyper-exponential	16	0.10	9.57	18.78
Pareto-2	∞	0.10	9.62	18,85

To summarize, the results in Tables I to III demonstrate that (a) $\mathbb{E}[S_0]$ is nearly insensitive to the job-size distribution for a wide variety of parameter settings, and (b) calculations of the optimal performance based can be calculated very accurately based on (1)-(6).

IV. Score-Function Based Splitting Method

Our splitting-and-merging method the functionality is implemented between the transport layer and the application

layer, and is based on the following score function (which was introduced in [29]) that is repeatedly measured for each of the N parallel TCP-connections: for $i = 1, \ldots, N$,

$$score_i = \frac{Q_i}{G_i} + \frac{sRTT_i}{2}. \quad (7)$$

Here, Q represents the length in bytes of the data that has to be transmitted, and $sRTT$ is the smoothed RTT that each TCP implementation estimates. G is the smoothed throughput (which is calculated for each individual connection), that is repeatedly estimated from the following update scheme: $G_0 := 0$, and for $t = 1, 2, \ldots$,

$$G_t = \alpha \times G_{t-1} + (1 - \alpha) \times T_t, \quad (8)$$

where the smoothing parameter α $(0 < \alpha < 1)$ is a constant, and T_t is the measured throughput (for each individual connection) which is continuously measured every τ milliseconds. The parameter τ balances the trade-off between computational resources for calculating the score-function periodically and the responsiveness to changes in the RTT and the throughput. The score function (7) estimates the time of arrival of TCP segments at the receiver re-sequencing point by accounting for its major delay factors at node i: (a) the queueing delay at the sender, estimated by the ratio Q_i/G_i, and (b) the transmission delay in the network, estimated by $sRTT_i/2$; in this way, the variations of the RTTs on the networks are accounted for implicitly.

Using the iterative scheme in (7) and (8), the method simply works as follows: After calculating the score function $score_i$ for each connection $i = 1, \ldots, N$, the method assigns the packet to the connection having the lowest score, effectively choosing the connection that has the lowest estimated delay up to point where all traffic is merged back.

The score-function method (7)-(8) was introduced earlier by Hasegawa et al. [29], called the Arrival-Time matching Load-Balancing (ATLB) method. However, our implementation of the method leads to a number of important benefits. First, the authors in [29] propose to modify the TCP protocol and to deploy the *implementation in a separate gateway* to avoid the difficulties of replacing parts of the operating system. Instead, our implementation overcomes these difficulties by executing the scheduling functionality in application-space of a Linux operating system with the same API towards the existing applications and using the standard TCP sockets interface for the multi-path sessions. Second, as pointed out in [29] TCP througput degradation may occur in real network environments because the receiving buffer for sorting the data segments is of limited size. As described in [8] high data rate differences between multiple networks may cause the advertised window of the faster TCP session to reach zero because the packets from the faster connection will fill the buffer and force TCP to slow-down. Although our implementation uses a limited receiving buffer, the aforementioned TCP throughput degradations do not occur. Third, the ATLB implementation [29] requires a network proxy, which raises the need for having a separate proxy for each access network. This is undesirable because the very goal of traffic splitting is to optimize performance over those access networks. Our method, instead, overcomes this problem because it does not require a network proxy, and hence can be installed both in an end-node device and in a network proxy. These observations make our method much more easy-to-deploy in real networking environments.

V. Experimental Evaluation of Score-Function Based Approach

To assess the effectiveness and ease-of-use of the proposed approach, we have conducted test-lab experiments under representative circumstances, where the RTT is usually small and packet loss as observed by the transport layer is negligable [30]. This section summarizes the results of extensive lab experiments in which two independent access networks form a CA network, which represents the most common scenario. Our aim is to demonstrate that (a) optimal traffic splitting can be closely approached using real networks with a contemporary traffic splitting solution, and (b) our model can be applied for predicting the flow-level performance of contemporary traffic splitting solutions. To this end, we have implemented the score-function method in a realistic test-lab environment. Using this implementation, the experimental results were validated against the benchmark results obtained from (1)-(6). In Section V-A we discuss the experimental setup, and in Section V-B we give an outline of the results.

A. Experimental setup

To perform lab experiments, we have connected two powerful multi-homed PCs by two independent and similar access networks to measure the download response times of foreground file transfers in the presence of (file transfer) background traffic and compare the obtained values with the expected file download time from the model. The experimental setup is depicted in Figure 2, where the PC with the FTP clients generates all FTP-download requests according to independent Poisson processes, for the foreground and both background streams. It is important to state that, even with a larger number of WLAN client devices in addition to the access point (AP), only one station is contending for the medium at the same time, as reported in [31] and observed during the experiments in [17]. Therefore, the use of only one client device yields outcomes that are representative for larger client populations.

At the PC serving as FTP server, 40,000 files have been generated according to an exponential distribution (with mean size 1×10^6 bytes) that is superimposed on a deterministic one 1×10^6 bytes to obtain a total mean file size of 2×10^6 bytes. The motivation to have at least a file size of 1×10^6 bytes is that it takes some time (and bytes) to have the splitting method operating properly and thus a bias against short files can be expected that is stronger than the one from a single TCP connection. Finally, during the experiments the files to be downloaded are randomly selected by the client PC during the experiments.

In [17] a PS-model was validated that was able to accurately model the file transfer response time over a WLAN. In this PS-model the highly complex dynamics of the FTP/TCP/IP/MAC-stack, and their interactions, are translated into a single parameter, called the *effective load*. The effective load, denoted ρ_{eff}, is subsequently used to describe the flow-level

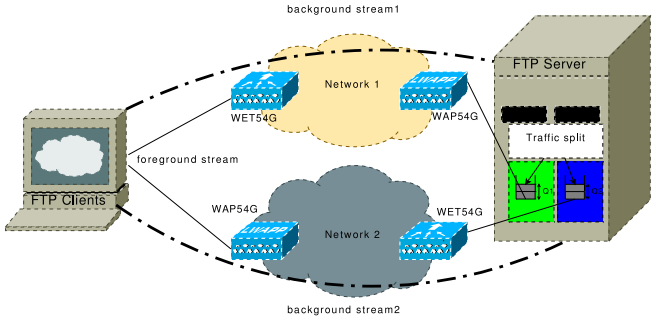

Fig. 2: Experimental setup.

TABLE IV: Test-lab environment and model parameters.

Parameter	Value	Parameter	Value
mac	224 bits	ack	112 bits
τ	20μs	difs	50μs
Cw_{min}	31 (slots(τ))	sifs	10μs
phy	192μs	eifs	364μs
δ	1μs	rwlan	11Mbps
X_{FTPget}	4096 bits	$X_{FTPclosure}$	64 bits
X_{MSS}	11680 bits	$X_{tcp/ip}$	320 bits
w	8760 bytes	X_{file}	2×10^6 bytes

behavior of FTP-based file transfers over WLANs without admission control as an M/G/1 Processor Sharing (PS) - model. Hence, we use the model from [17] to parameterize both our test-lab environment and the model to assess the efficiency of a TCP-based traffic splitting solution in the test-lab using real (WLAN) networks. In accordance to the model conditions in [17] we have configured our test-lab environment similarly with the same TCP window size, a sufficiently large AP buffer to avoid overflows, and restricted the maximum load per network (to values below $\rho_{eff} = 0.88$). Under these circumstances, it was shown that the model leads to highly accurate predictions over a wide range of parameters combinati ons, including light- and heavy-tailed file-size distributions and light- and heavy-load scenarios. In addition, the observed mean download response times are under these circumstances fairly insensitive to the file-size distribution, as suggested by the PS-model.

In our test-lab, each wireless access network consists of a Linksys (WAP54G) access point and a wireless Ethernet bridge (WET54G) of the same hardware and firmware version connected by a power splitter to avoid interference. The AP uses a modified firmware program, called OpenWrt, that is specifically designed for embedded devices such as residential gateways and routers. This firmware offers detailed WLAN-MAC configuration options, combined with a sufficiently large AP buffer necessary for our experiments. Table IV summarizes the WLAN configuration that is used in our test-lab, and the corresponding model to assess the performance of our practical solution. In Table IV the MAC overhead bits, mac, the acknowledgment overhead bits, ack, the minimum congestion window size, Cw_{min}, the slot time, τ, and the inter-frame spacing times (sifs, difs,eifs) match with a default IEEE 802.11b MAC operating in basic access mode. The WLAN transmission rate of the data and acknowledgment frames is represented by $rwlan$ and set to 11Mbit/s. Furthermore, the preamble of the Physical Layer Convergence Protocol (PLCP), indicated phy, was set to long. Subsequently, the TCP stack in our test-lab environment was configured to use an MSS, indicated as X_{MSS}, of 1460 bytes and a window size, w, of 8760 bytes. Therefore we use in our model 40 bytes of TCP/IP overhead, represented by variable $X_{tcp/ip}$. Moreover, the FTP application is assumed to use an FTP GET command (assumed equally sized as trivial ftp (tftp) requests) with a size of 512 bytes or 4096 bits(X_{FTP}) and an FTP closure command, $X_{FTPclosure}$, of 64 bytes. Finally, X_{file} denotes the average file size used in our experiments. Based on the parameters in Table IV, we obtain from the model in [17] an overall effective throughput, r_{dwnld}, of 5.28Mbps. In the effective throughput all protocol (MAC/IP/TCP/FTP) overhead and dynamics is accounted and (validated under respresentative circumstances) and corresponds to the expected throughput rate received by the file download. Based on the overall effective throughput, the expected service time of a file in one network is equal to $\beta = r_{dwnld}/X_{file}$ and in our case $\beta = 3.03$ seconds.

To make the adjustment to changing network conditions very responsive, in our experiments the smoothing parameter α of the score-function, presented in (8) was set to 0.4. Hasegawa et al. [29] argue that the measurement interval τ should be fixed and set to a value of 100ms for network with an RTT of 20 ms. However, test-lab experimentation (results are not presented here) shows that it is better to adapt τ to changing RTT values in the network. More precisely, a best practice rule-of-thumb is to set τ equal to $4 \times RTT$ because a shorter period does not give a good performance estimation (as fast retransmissions and the OS scheduling accounted for sufficiently).

B. Experimental results

In this section we report on the experiments conducted on our test-lab environment to determine efficiency of the TCP-based splitting solution in a real networking environment. The model for predicting the file download times consists of two components; the first component is the WLAN PS-model from [17] that is able take the WLAN specific parameters into account to determine the effective load of the network and the expected download time in one PS node. Secondly, based on the effective service time of a file download, we obtain the optimal dynamic splitting performance over two networks by parameterizing the splitting model presented in Section 2 (see also Remark 2 above).

For our experiments we have varied the effective load according to Tables V and VI for the foreground traffic ρ_0, and the background traffic in network one and two, ρ_1 and ρ_2, respectively. From the effective loads we determined the download request arrival rates for the FTP clients. In Tables V and VI, the 15 runs that we have performed are shown with their respective effective load values. To obtain the specified 95% confidence intervals for the measured average download response time, denoted $E[S_i]_{exp}$, the experiments required an execution time from 2-10 days per run. Several representative results are shown in Tables V and VI. More precisely, for a variety of (ρ_0, ρ_1, ρ_2) combinations Table V

TABLE V: Experimental and analytical values for $\mathbb{E}[S_0]$ for different foreground and background load combinations.

			$\mathbb{E}[S_0]$		
ρ_0	ρ_1	ρ_2	exp	opt	$\Delta\%$
0.1	0.1	0.5	2.268	2.145	5.7%
0.1	0.2	0.3	2.051	2.029	1.1%
0.1	0.3	0.7	3.008	2.914	3.2%
0.1	0.4	0.5	2.693	2.646	1.8%
0.1	0.4	0.6	3.053	2.913	4.8%
0.1	0.5	0.5	3.043	2.881	5.6%
0.5	0	0.8	3.965	3.787	4.7%
0.5	0.1	0.2	2.559	2.419	5.8%
0.5	0.2	0.3	2.932	2.788	5.2%
0.5	0.3	0.7	5.259	4.922	6.9%
0.5	0.5	0.5	4.994	4.900	2.0%
0.5	0.5	0.6	6.069	5.975	1.6%
0.9	0.2	0.3	4.782	4.526	5.7%
0.9	0.3	0.4	6.589	6.426	2.5%
1.2	0	0	3.952	3.870	2.1%

TABLE VI: Experimental and analytical values for $\mathbb{E}[S_1]$ and $\mathbb{E}[S_2]$ for different foreground and background load combinations.

			$\mathbb{E}[S_1]$			$\mathbb{E}[S_2]$		
ρ_0	ρ_1	ρ_2	exp	opt	$\Delta\%$	exp	opt	$\Delta\%$
0.1	0.1	0.5	3.614	3.575	1.1	6.622	6.422	3.1
0.1	0.2	0.3	4.049	4.006	1.1	4.561	4.578	-0.4
0.1	0.3	0.7	4.763	4.714	1.0	11.061	11.036	0.2
0.1	0.4	0.5	5.437	5.438	0.0	6.435	6.524	-1.4
0.1	0.4	0.6	5.535	5.481	1.0	8.484	8.216	3.3
0.1	0.5	0.5	6.619	6.569	0.8	6.676	6.568	1.6
0.5	0	0.8	0.00	0.000	0.0	23.007	23.721	-3.0
0.5	0.1	0.2	4.496	4.439	1.3	5.101	4.979	2.5
0.5	0.2	0.3	5.330	5.219	2.1	5.985	5.946	0.7
0.5	0.3	0.7	7.809	7.539	3.6	17.103	17.297	-1.1
0.5	0.5	0.5	10.529	10.399	1.3	10.620	10.400	2.1
0.5	0.5	0.6	11.734	11.537	1.7	14.254	14.369	-0.8
0.9	0.2	0.3	8.194	8.154	0.5	9.637	9.241	4.3
0.9	0.3	0.4	11.866	11.759	0.9	13.565	13.598	-0.3

shows the experimental and the simulated values of $\mathbb{E}[S_0]$ (denoted $\mathbb{E}[S_0]_{exp}$ and $\mathbb{E}[S_0]_{opt}$, respectively), and their relative difference, defined as by

$$\Delta\% := \frac{\mathbb{E}[S_0]_{exp} - \mathbb{E}[S_0]_{opt}}{\mathbb{E}[S_0]_{opt}} \times 100\%. \quad (9)$$

The width of the 95% confidence intervals is less than 5% in all cases. Note that to obtain the model outcomes for the expected foreground and background download response time, $E[S_0]$, the Markov model from Section II (Remark 3) is numerically solved for the corresponding effective-load values, with mean processing time $\beta_1 = \beta_2 = 3.03$ seconds. Similarly, Table VI shows the results for the performance of the background traffic streams, $\mathbb{E}[S_1]$ and $\mathbb{E}[S_2]$.

Based on the outcomes in Tables V and VI, it can be concluded that near-optimal performance can be achieved by our score-function based algorithm for routing TCP segments. The differences between the benchmark and the measured performance are very small, and typically no more than a few percent. The smallest difference between the benchmark performance and performance in practice is obtained for the smallest traffic intensity, which may be explained by the smaller number of adaptations required of the splitting method in comparison to other runs in which one of the networks has a higher effective load.

One run that exemplifies the good performance of CA is the one in the third row of Table V, where $\rho_0 = 0.1$, $\rho_1 = 0.3$ and $\rho_2 = 0.7$. Here, the measured average download response time is even lower than its service demand of 3.03 seconds, even in the presence of an effective load of 1.1 on a total system capacity of 2.

Some comments are due. First, it should be noted that the splitting model used as the basis for the performance benchmark assumes exponentially distributed service times, whereas the test-lab uses a slightly different file-size distribution. However, taking the results into account from Section III the error due to this can be regarded limited in view of the small differences between sojourn times obtained from the Pareto-2 and deterministic distributions. Second, given the limited sensitivity of the sojourn time to the job-size distribution, the test-lab results are also representative for other file-size distributions, as long as a minimum size is respected that is comparable to the one we used in our experiments. Third, for experiments in which one of the networks was heavily loaded ρ_1, $\rho_2 \geq 0.8$, the differences are getting large. This may be explained by the higher number of capacity fluctuations that may be perceived by the TCP sessions to which the traffic distribution may need to respond. Another cause of this larger difference may be the inaccuracy of the WLAN PS-model from [17] that was validated up to an effective load of 0.88, and shown to gradually underestimate the download response times for such high load values and beyond.

Hasegawa et al. [29] also report valuable but quite limited experimental results on the score-function method. However, there are some important differences. First, in our experiments the background traffic is assumed to be highly dynamic. In our experiments background jobs arrive according to Poisson processes, and the the job-size distributions may be highly variable, whereas the background streams in [29] are persistent and hence do not require a highly adaptable traffic-splitting algorithm. Second, the experiments in [29] are focused on capacity aggregation and on optimizing throughput over parallel networks. On the contrary, our analysis and experiments are oriented towards minimizing expected response times. Moreover, our measurements allow us to quantify the efficiency of our splitting algorithm by comparing the results with the benchmark results from our analytical model.

VI. Conclusions

This paper describes an analytic model that can be used to determine the optimal traffic splitting performance by solving a CMTC. It is shown that this optimal performance can be closely approached with a remarkably simple and easy-to-deploy score-function based algorithm that dynamically splits TCP segments based on on-the-fly measurements on the per-node RTT, the sending-buffer contents and the estimated throughput. We believe that this forms a significant step towards filling the gap between the theoretical modelling and analysis of traffic splitting among multiple network paths and practice.

VII. Topics for Further Research

The results presented in this paper lead to a variety of challenges for follow-up research. First, as pointed out in [17], the accuracy of PS models becomes questionable under high traffic loads or system under-dimensioning so that (bursty) packet loss has a significant impact on the results. It needs further exploration how the parameterizations in (7) and (8) need to be adjusted, e.g. by *adapting* the smoothing parameter α and the measurement interval τ, in response to bursty packet loss patterns.

Second, the CAN-model is based on the assumption that the traffic is elastic, in the sense that the available bandwidth changes over time caused by fluctuations in the number of simultaneous flows in the network. However many applications (such as streaming applications) are not elastic, and instead require fixed amounts of network bandwidth. To incorporate smart traffic splitting for mixtures of elastic and streaming applications, the model should be extended to mixtures of fixed and elastic traffic streams. Optimal splitting in such multiclass models opens up a challenging area for follow-up research.

Third, the fluid-based dynamic splitting of foreground jobs (as discussed in Section 2) has the attractive property that the synchronization delay is zero, while at the same time the policy is fair in the sense that foreground jobs receive no more than their fair share of the capacity at each of the individual nodes. It seems obvious that even better performance for the foreground traffic can be obtained by dropping the restriction of PS-based fair sharing of capacity for each node, and allowing for the possibility of some kind of weighted sharing of the capacity of the nodes in favor of the foreground traffic, which obviously comes at a price to be paid by the background traffic streams. Further optimization of the performance of foreground streams, and balancing the trade-off with the degradation of the performance of the background streams opens up an interesting direction for further research.

Fourth, in [28] a tail-optimal traffic splitting rule was formulated for static (i.e. the file is split once, according to fixed portions upon arrival) traffic distribution under light foreground load. For dynamic traffic splitting, the splitting ratio is an output variable rather than an input. It remains an interesting topic for further research how the splitting ratios obtained from dynamic traffic splitting relate to a splitting rule that performs well for a static splitting policy.

Finally, another topic of further research is to study the impact of the arrival process on the download response times, if for instance an MMPP arrival process is applied instead of Poisson. Our study has so far concentrated on the foreground traffic performance. One may expect that also the background traffic may benefit from the dynamic splitting of foreground traffic, as it will send less traffic to the networks with the highest instantaneous loads.

VIII. Acknowledgments

This work was performed within the project RRR (**R**ealisation of **R**eliable and Secure **R**esidential Sensor Platforms) of the Dutch program *IOP Generieke Communicatie*, number IGC1020, supported by the *Subsidieregeling Sterktes in Innovatie*. The contribution of R.D. van der Mei has been partially funded by the Dutch Ministry of Economic Affairs, project "Service Optimization and Quality" (IGC0820).

References

[1] D. Cox, "Fundamental limitations on the data rate in wireless systems," *IEEE Communications Magazine*, vol. 46, no. 12, pp. 16–17, 2008.

[2] FCC, "Report of the spectrum efficiency working group," Federal Communications Commission Spectrum Policy Task Force, Tech. Rep., November 2002.

[3] J. Duncanson, "Inverse multiplexing," *IEEE Communications Magazine*, vol. 32, no. 4, pp. 34–41, 1994.

[4] I. S. 802.11n, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer specifications Enhancements for Higher Throughput," October 2009.

[5] R. Chandra, P. Bahl, and P. Bahl, "Multinet: Connecting to multiple IEEE 802.11 networks using a single wireless card," in *Proceedings IEEE INFOCOM 2004*, 2004.

[6] G. Koudouris, R. Agüero, E. Alexandri, J. Choque, K. Dimou, H. Karimi, H. Lederer, J. Sachs, and R. Sigle, "Generic link layer functionality for multi-radio access networks," in *Proceedings* 14^{th} *IST Mobile and Wireless Communications Summit*, 2005.

[7] F. Berggren and R. Litjens, "Performance analysis of access selection and transmit diversity in multi-access networks," in *MobiCom '06: Proceedings of the* 12^{th} *Annual International Conference on Mobile Computing and Networking*, New York, NY, U.S.A., 2006, pp. 251–261.

[8] H. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," *Wireless Networks*, vol. 11, no. 1, pp. 99–114, 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-004-4749-6

[9] J. Iyengar, P. Amer, and R. Stewart, "Concurrent multipath transfer using sctp multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, 2006.

[10] J. Iyengar, "End-to-end concurrent multipath transfer using transport layer multihoming," Ph.D. dissertation, University of Delaware, 2006.

[11] C. Huang and C. Tsai, "The handover control mechanism for multi-path transmission using stream control transmission protocol (SCTP)," *Computer Communications*, vol. 30, no. 17, pp. 3239–3256, 2007.

[12] R. Stewart, "Stream control transmission protocol," Internet Engineering Task Force, RFC 2960, October 2000.

[13] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, Berkeley, CA, U.S.A., 2004, pp. 99–112.

[14] S. Borst, O. Boxma, and N. Hegde, "Sojourn times in finite-capacity processor-sharing queues," in *Proceedings NGI 2005 Conference*, 2005.

[15] Y. Wu, C. Williamson, and J. Luo, "On processor sharing and its applications to cellular data network provisioning," *Performance Evaluation*, vol. 64, no. 9-12, pp. 892–908, 2007.

[16] R. Litjens, F. Roijers, J. van den Berg, R. Boucherie, and M. Fleuren, "Performance analysis of wireless LANs: an integrated packet/flow level approach," in *Proceedings of the* 18^{th} *International Teletraffic Congress - ITC18*, Berlin, Germany, 2003, pp. 931–940.

[17] G. Hoekstra and R. van der Mei, "Effective load for flow-level performance modelling of file transfers in wireless LANs," *Computer Communications*, vol. 33, no. 16, pp. 1972–1981, 2010.

[18] P. Key, L. Massoulié, and D. Towsley, "Combining multipath routing and congestion control for robustness," in *Proceedings Conference on Information Sciences and Systems*, 2006.

[19] ——, "Path selection and multipath congestion control," in *Proceedings IEEE INFOCOM 2007*, 2007, pp. 143–151.

[20] S. Ko and R. Serfozo, "Response times in M/M/s fork-join networks," *Advances in Applied Probability*, vol. 36, no. 3, pp. 854–871, 2004.

[21] G. Choudhury and D. Houck, "Combined queuing and activity network based modeling of sojourn time distributions in distributed telecommunication systems," in *Proceedings of the* 14^{th} *International Teletraffic Congress - ITC14*, Antibes, France, 1994, pp. 525–534.

[22] L. Flatto and S. Hahn, "Two parallel queues created by arrivals with two demands," *SIAM Journal on Applied Mathematics*, vol. 44, no. 5, pp. 1041–1053, 1984. [Online]. Available: http://link.aip.org/link/?SMM/44/1041/1

[23] J. Lui, R. Muntz, and D. Towsley, "Computing performance bounds for fork-join queueing models," The Chinese University of Hong Kong, Tech. Rep., 1994.

[24] M. Lelarge, "Packet reordering in networks with heavy-tailed delays," *Mathematical Methods of Operations Research*, vol. 67, no. 2, pp. 341–371, 2008.

[25] ——, "Tail asymptotics for discrete event systems," in *Proceedings Valuetools '06:* 1^{st} *international conference on Performance evaluation methodolgies and tools*, 2006, pp. 563–584.

[26] E. Altman, U. Ayesta, and B. Prabhu, "Load balancing in processor sharing systems," in *Proceedings of the* 3^{rd} *International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools '08, 2008, pp. 12:1–12:10.

[27] F. Baccelli, W. Massey, and D. Towsley, "Acyclic fork-join queuing networks," *Journal of the ACM*, vol. 36, no. 3, pp. 615–642, 1989.

[28] G. Hoekstra, R. van der Mei, Y. Nazarathy, and A. Zwart, "Optimal file splitting for wireless networks with concurrent access," *Lecture Notes in Computer Science*, vol. 5894, pp. 189–203, 2009.

[29] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase, "Deployable multipath communication scheme with sufficient performance data distribution method," *Computer Communications*, vol. 30, no. 17, pp. 3285–3292, 2007.

[30] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, pp. 6–16, 2003.

[31] F. Roijers, J. van den Berg, and X. Fang, "Analytical modelling of TCP file transfer times over 802.11 wireless LANs," in *Proceedings of the* 19^{th} *International Teletraffic Congress - ITC19*, Beijing, China, 2005.