Efficient Traffic Splitting over Parallel Wireless Networks with Partial Information

Up to now, we have considered information flows over a single path through the network. Packet-switched networks offer the possibility to exploit several paths in parallel. Multi-path communication solutions provide a promising means to improve network performance in areas covered by multiple wireless access networks. Today, little is known about how to effectively exploit this potential. In this chapter we study a model where jobs are transferred over multiple parallel networks, each of which is modeled as a processor sharing node. The goal is to minimize the expected transfer time of elastic data traffic by smartly dispatching the jobs to the networks, based on partial information about the numbers of foreground and background jobs in each of the nodes. In the case of full state information, the optimal policy can be derived via standard MDP-techniques, but for models with partial information an optimal solution is hard to obtain. An important requirement is that the splitting algorithm is efficient, yet simple, easy-to-implement, scalable in the number of parallel networks and robust against changes in the parameter settings. We propose a simple index rule for splitting traffic streams based on partial information, and benchmark the results against the optimal solution in the case of full state information. Extensive simulations with real networks show that this method performs extremely well under practical circumstances for a wide range of realistic parameter settings.

4.1 Background

Today, many wireless networks have already closely approached the Shannon limit on channel capacity, leaving complex signal processing techniques room for only modest improvements in the data transmission rate [40]. A powerful alternative to increase the overall data rate then becomes one in which multiple, likely different, networks are used concurrently because (a) the spectrum is regulated among various frequency bands and corresponding communication network standards, and (b) the overall spectrum usage remained to be relatively low over a wide range of frequencies [46]. The concurrent access to multiple networks simultaneously has opened up enormous possibilities for increasing bandwidth, improving reliability, and enhancing Quality of Service (QoS) in areas that are covered by multiple wireless access networks. Currently, the efficient use of multiple networks concurrently is an active area of both research [97] and standardization efforts [54]. However,

\[^{4}\text{This chapter is based on [19] and [14].}\]
despite the enormous potential for quality improvement, only little is known about how to fully exploit this potential. This raises the need for new splitting algorithms for concurrent access that are simple, easy to implement, yet effective.

The effectiveness of splitting data traffic streams is generally believed to increase when detailed information about the state of the system (e.g., the number of flows, measured round-trip times and the network load) is available. In practice, however, there is often no such detailed information available, or at best only some coarse-grained and aggregated statistics. Consequently, a key the challenge is to achieve efficient network utilization levels and good end-user application performance, based on information that is only partially available. At the same time, for the practical usefulness the splitting algorithms are required to be simple, easy-to-implement, scalable in the number of access networks and robust against changes in the parameter settings. We emphasize the importance of this requirement: 'optimal' splitting algorithms based on idealized situations where all detailed information is available – if possible at all – are often too complicated to be practically infeasible.

Processor Sharing (PS) models provide a powerful means to model the bandwidth sharing behavior of elastic traffic streams in TCP-based data networks. A particularly attractive feature of these models is that they abstract from the complex packet-level details of the network, but at the same time maintain the essential factors that determine the data transfer–time performance of elastic data flows. Moreover, the theory of PS models is well-matured and has been successfully applied to model the flow–level behavior of a variety of communication networks, including CDMA 1xEV-DO [18], WLAN [59], UMTS–HSDPA [109] and ADSL [9]. In [59], an analytic flow–level model was presented that explicitly translates the complex and detailed packet-level dynamics of the FTP/TCP/IP-stack over a WLAN into a M/G/1–PS model for the flow–level performance of data transfers.

In the literature, a variety of fundamental and applied studies have been focused to the splitting and scheduling jobs to multiple nodes. The available results and techniques are outlined below. In the context of telecommunication systems, the concurrent use of multiple network resources in parallel was already described for a Public Switched Digital Network (PSDN) [43], where inverse multiplexing was proposed as a technique to perform the aggregation of multiple independent information channels across a network to create a single higher–rate information channel. Various approaches have appeared to exploit multiple transmission paths in parallel. For example, by using multi–element antennas, as adopted by the IEEE 802.11n standard [64], at the physical layer or by switching datagrams at the link layer [31, 72], and also by using multiple TCP sessions in parallel to a file server [95]. In the latter case, each available network transports part of the requested data in a separate TCP session. Previous work has indicated that downloading from multiple networks concurrently may not always be beneficial [52], but in general significant performance improvements can be realized [56, 58, 62]. Under these circumstances of
4.1 Background

using a combination of different network types, in particular, the transport layer-
approaches, have shown their applicability [62] as they allow appropriate link layer
adaptations for each TCP session.

In a queueing-theoretical context, only few papers study partial information mod-
els. Bellman [10] was the first to study decision problems with a transition law that is
not completely known. He observed that the problem could be transformed into an
equivalent full observation problem by augmenting the state space with the set of
probability distributions defined on the domain of the unknown quantity (i.e., the un-
observed state, or the unknown parameter) and updating it by Bayes’ rule. The trans-
formation of the partial information problem to the complete information model,
however, comes with added computational difficulties, since policies are defined
over a continuum of states. This is the fundamental problem in developing algo-
rithms for computing optimal policies [89]. There is some work in the theoretical
domain to characterize the structure of the optimal policy (see, e.g., [25, 2, 103, 79]).
Even then, finding the optimal policy computationally for a general Bayesian deci-
sion problem is intractable. Approaches dealing with this are to be satisfied with
suboptimal solutions or to develop algorithms that can exploit problem characteris-
tics (see, e.g., [78, 90, 116, 57, 24, 30]). We refer to [80, 85, 107, 76] for some surveys
on computational techniques.

In this chapter we study a model (depicted in Figure 4.1) consisting of $N$ non-identical
parallel networks that are modeled as PS nodes that serve $N + 1$ streams of jobs.
Node $i$ has processing speed $C_i$. Stream 0 is called the foreground stream, and
streams 1, ..., $N$ are called the background streams. Jobs of background stream $i$
are served exclusively at PS node $i$. Each job of the foreground stream has to be
routed to exactly one of the PS nodes by the dispatcher. Job sizes are assumed to be
exponentially distributed. The goal is to develop a dynamic dispatching policy that
minimizes the expected sojourn time of foreground jobs by using information about
the numbers of foreground and background jobs at each of the PS nodes. Based
on practice, we assume that the dispatcher is not able to distinguish the number of
foreground and background jobs in the network, but instead only has information
about the total number of jobs.

The model under consideration (see Section 4.2 below for details) was also stud-
ied in [14], where we addressed this problem through a learning mechanism, where
the dispatcher makes a statistical inference on the distribution of the numbers of
foreground and background jobs after the each decision. This Bayesian splitting al-
gorithm in [14] was found to be highly effective in dealing with partial information,
and its performance was found to be close to the performance of the optimal pol-
cy under full state information. However, the Bayesian approach has two main
drawbacks: (1) the method is quite complicated and requires in-depth knowledge
about stochastic models, which limits its practical usefulness, and (2) the method
is not scalable in the number of parallel access networks, $N$, because it needs the
full-state information MDP solution, which suffers from the curse of dimensionality. This limits the applicability of the Bayesian methods due to memory constraints.

The method presented in this chapter is a simple index rule that is essentially a convex combination of techniques that are found to work well extreme cases: (1) the Weighted Join Shortest Queue (WJSQ) policy that routes foreground flow arrivals to the node where the total number of flows, normalized by the node speed, is minimized, and (2) the Conditional Sojourn Time (CST) approach where the expected sojourn time, conditioned on the total numbers of flows at each of the networks, is minimized. The WJSQ policy is particularly effective when the foreground traffic load tends to saturate the nodes, whereas the CST policy is expected to perform well in systems with low load, or low foreground load situations. The interpolating factor, denoted $\alpha$ ($0 < \alpha < 1$), represents the ratio of the foreground load and the remaining amount of capacity. When $\alpha \approx 0$ the CST policy is expected to perform well, whereas for $\alpha \approx 1$ the WJSQ policy is expected to perform well. To assess the effectiveness of the CC method, we have performed extensive simulation experiments in a real network simulator, called OPNET $[87]$, that implements the full wireless protocols stack. The results show that the CC method leads to close-to-optimal performance for a wide range of realistic parameter settings.

We emphasize that the main contribution of this chapter lies in (1) its practical usefulness, providing a simple but very effective means to (near-)optimally split elastic traffic streams over wireless networks based on limited information about the state of the system, (2) its scalability with respect to the number of parallel access networks, and (3) the fact the efficiency of the splitting approach is extensively validated by a wide range of real network simulations (rather than simplified queueing simulations) implementing the complex dynamics of full wireless protocol stacks.

The organization of the chapter is as follows. In Section 4.2 we describe the model and introduce the notation. In Section 3 we discuss the full-state information model and present our simple index-rule based heuristic. In Section 4.4 we discuss the results of extensive numerical evaluation of the heuristic in realistic network simulations with OPNET $[87]$, where full wireless protocol stack is implemented.

### 4.2 Model

We study a model consisting of $N$ non-identical parallel networks that are modeled as PS nodes that serve $N + 1$ streams of flows (we refer to $[59]$ for details on the validation and the parameterization of PS models for modeling wireless networks). Stream 0 is called the *foreground* stream, and streams $1, \ldots, N$ are called the *background* streams. From each stream flows arrive according to a Poisson process with arrival rate $\lambda_i$, ($i = 0, 1, \ldots, N$). Flows from background stream $i$ are served exclu-
sively at PS node \( i \). Each flow from the foreground stream has to be dispatched to one of the PS nodes on the basis of information on the \textit{total} number of flows (thus, number of foreground flows \textit{plus} the number of background flows) at each of the nodes, such that the expected sojourn time \( E[S_0] \) for an arbitrary foreground flow is minimized. Flow sizes are assumed to be exponentially distributed with rate \( \mu \), and each node has processing speed \( C_i \), so that server \( i \) can handle \( C_i \mu \) flows per time unit. Without loss of generality, the node capacities are normalized such that \( C_1 + \cdots + C_N = 1 \). For each node \( i \), the offered background load is given by \( \rho_i := \lambda_i / C_i \mu \) (\( i = 1, \ldots, N \)), and the foreground load is \( \rho_0 := \lambda_0 / \mu \). Considering all arriving flows, the total offered load is given by \( \rho := \rho_0 + \sum_{i=1}^{N} \rho_i C_i \). For stability reasons, we assume \( \rho < 1 \). The fraction foreground load compared to the total load is denoted by \( \beta := \rho_0 / \rho \).

In general, for each given splitting policy that bases its routing decision on the full state information, the model can be described as a CTMC with state space \( S = \mathbb{N}_0^2 \times \mathbb{N}_0^N \), where \( \mathbb{N}_0 \) is the set of nonnegative integer numbers. Each state \( s \in S \) can be written as \( s = (x_1, \ldots, x_N, y_1, \ldots, y_N) \), with \( x_i \) the number of foreground flows on the nodes and \( y_i \) the number of background flows. In this chapter, it is assumed that the dispatcher has only access to \textit{partial information} in the sense that it has knowledge of \( z_i := x_i + y_i \) for \( i = 1, \ldots, N \), i.e. the \textit{total} number of flows on each of the nodes. Recall that in the case of full state information the dispatcher has knowledge of \( (x_1, \ldots, x_N, y_1, \ldots, y_N) \). Based on the above information, there is a central decision maker that has to decide on the distribution of the foreground jobs over the \( N \) servers. In doing so, the aim is to have a decision policy that minimizes \( E[S_0] \), where \( S_0 \) is the sojourn time of an arbitrary foreground job in the system.
Chapter 4: Efficient Traffic Splitting over Parallel Wireless Networks with Partial Information

4.3 Splitting algorithms

In this section we describe a number of splitting algorithms, which will be evaluated in the next section. In 4.3.1 we describe the MDP model for the case of full state information (see [94] for details on MDP’s), which will be used as a benchmark to assess the efficiency of the index rule for the case of partial information. In Section 4.3.2 we describe both the Bayesian partial information approach 4.3.2.1 and our index rule 4.3.2.2.

4.3.1 Full state information

In this subsection we assume that the dispatcher has full state information, and formulate the optimal dispatching problem as a Markov decision process (MDP). More specifically, the dispatching decisions are based on state description \( s = (x_1, \ldots, x_N, y_1, \ldots, y_N) \) where \( s \in S \) is from the state space \( S = \mathbb{N}_0^{2N} \), and where \( \mathbb{N}_0 \) is the set of nonnegative integers. Let \( \mathcal{I} = \{1, \ldots, N\} \) be the set of nodes and \( \mathcal{A} = \{1, \ldots, N\} \) be the set of actions, where the action \( i \) means that the dispatcher forwards the flow to node \( i \in \mathcal{A} \). The goal is to minimize the total expected response time by minimizing the total number of active foreground flows. Note that the MDP will not directly obtain the expected sojourn time for foreground flows but by using Little’s Law, \( \lambda_0 \mathbb{E}[S_0] = \mathbb{E}[N_0] \), we can obtain the average response time from \( \mathbb{E}[N_0] \), the average number of foreground flows. The reward function, corresponding to the total number of foreground flows, is defined as \( r(s) = x_1 + \cdots + x_N \), where \( s = (x_1, \ldots, x_N, y_1, \ldots, y_N) \in S \). Furthermore we assume that \( \lambda_0 + \cdots + \lambda_N + \mu = 1 \) we can always get this by proper scaling. Please note that we already assumed \( C_1 + \cdots + C_N = 1 \). Let \( V(s) \) be the value function, i.e., the asymptotic difference in total costs that results from starting the process in state \( s \) instead of some reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) \( g + TV = TV \), where \( T \) is the dynamic programming operator acting on \( V \) defined as follows (see [94] for details):

\[
TV(s) = \min_{i \in \mathcal{A}} \left\{ \lambda_0 V(s + e_i) \right\} 
+ \sum_{i=1}^{N} y_i + \lambda_i V(s + e_{i+N}) \tag{4.1a}
+ \sum_{i=1}^{N} \mu C_i \frac{x_i}{x_i + y_i} V(s - e_i) \tag{4.1b}
+ \sum_{i=1}^{N} \mu C_i \frac{y_i}{x_i + y_i} V(s - e_{i+N}). \tag{4.1d}
\]
4.3 Splitting algorithms

In optimality equation $TV(s)$ (4.1a) corresponds to the foreground flow arrivals that have to be optimized, (4.1b) corresponds to arrivals of cross-traffic flows, (4.1d) corresponds to departures of foreground flows, and (4.1c) corresponds to departures of cross traffic flows.

Applying the backward recursion results in an optimal policy $R^* \in A^{[S]}$. An optimal policy contains optimal decisions depending on the number of flows on the PS-nodes. For each state $s = (x_1, \ldots, x_N, y_1, \ldots, y_N)$, the policy found by the backward recursion $R^*(s) \in A$ will give an optimal action.

4.3.2 Partial information model

The dynamic server selection model with full information uses a state description $(x_1, \ldots, x_N, y_1, \ldots, y_N)$ with $2N$ entries. However, in practice, distinguishing the foreground traffic from the background traffic might not be feasible. In these cases, one can only observe the state $(z_1, \ldots, z_N)$ with $z_i = x_i + y_i$ for $i = 1, \ldots, N$. Now, the dynamic control policy that we derived in the previous section cannot be applied straightforwardly. We now describe two approaches to tackle this problem. First in Section 4.3.2.1 we describe the Bayesian partial information approach. Secondly in Section 4.3.2.2 we describe a near optimal approach based on the conditional sojourn times of the different nodes.

4.3.2.1 Bayesian partial information approach

To apply the control policy one needs to create a mapping from $(z_1, \ldots, z_N)$ to $(x_1, \ldots, x_N, y_1, \ldots, y_N)$, so that (an estimate of the) full information is recovered. Note that it is not sufficient to create a mapping solely based on $(z_1, \ldots, z_N)$ at each decision epoch, since it does not use the information contained in the sample path, i.e., many sample paths can lead to the same state $(z_1, \ldots, z_N)$. Therefore, we will use Bayesian learning that takes into account the complete history of states in the estimation procedure. We shall call $z = (z_1, \ldots, z_N) \in \mathbb{N}_0^N$ the observation state. In order to learn about the division between the number of foreground and background jobs, we will denote by $u_i(n)$ the probability that at server $i$ there are $n$ foreground jobs for $i = 1, \ldots, N$. The probability distribution $u_i$ will serve the purpose of information about the states that cannot be observed; hence, $u = (u_1, \ldots, u_N)$ is called the belief state. Note that the belief state space is of high dimension, namely $\prod_{i=1}^N \{u_i \in [0, 1]^{\mathbb{N}_0} \mid \sum_{x \in \mathbb{N}_0} u_i(x) = 1\}$.

Based on the observation and belief states, we construct a state space for the Bayesian dynamic program consisting of the vectors $s = (z, u)$. Note that every arrival and departure gives the system information on how to update the belief state.
Suppose that state $s$ is given and that an arrival of foreground job that is admitted to server $i$ occurs. The new state $s_{af_i}$ is then given by $s_{af_i} = (z + e_i, u')$ where $u'_i(x) = u_i(x-1)$ for $x > 0$ and $u'_i(0) = 0$, and where $u'_j(x) = u_j(x)$ for $j \neq i$. In case of arrival of a background job to server $i$, we have a new state $s_{ab_i} = (z + e_i, u)$.

In case of departures, we have a similar state transformation. When a foreground job leaves server $i$, then we have corresponding states $s_{df_i} = ([z-e_i]^+, u')$ with $u'_i(x) = u_i(x+1)$ for $x \geq 0$. Similarly, when a background job leaves server $i$, then we have $s_{db_i} = ([z-e_i]^+, u)$. Naturally, these transitions cannot be observed, so we take the expectation with respect to the probability distribution $u$ to average over all sample paths. This gives a new dynamic programming operator in which learning is incorporated. This is given by

$$TV(s) = \sum_{x_1 \in \mathbb{N}_0} \cdots \sum_{x_N \in \mathbb{N}_0} u_1(x_1) \cdots u_N(x_N) \left[ \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} \lambda_i V(s_{ab_i}) \right.$$

$$\left. + \lambda_0 \min\{V(s_{af_1}), \ldots, V(s_{af_N})\} \right)$$

$$\left. + \sum_{i=1}^{N} \frac{x_i}{z_i} \mu_0 V(s_{df_i}) + \sum_{i=1}^{N} \frac{z_i-x_i}{z_i} \mu_i V(s_{db_i}) \right. + \left(1 - \lambda_0 - \sum_{i=1}^{N} \left[ \lambda_i + \frac{x_i}{z_i} \mu_0 + \frac{z_i-x_i}{z_i} \mu_i \right] \right) V(s). \right]$$

Note that the basic idea to transform Equation (4.1) into Equation (4.2) is to take the conditional expectation with respect to the belief state distribution $u$. Under this condition, the foreground and background jobs can be distinguished so that the structure of the equation resembles the one of the fully observed problem. However, only the transitions to the new belief state need to be adjusted so that the information that has been learned is taken into account. These transitions are provided above.

We end this section with two remarks.

4.3.1. Remark (Complexity). Note that the dynamic programming operator for the Bayesian model (4.2) resembles the dynamic programming operator of the full observation model (4.1). However, the state space of the Bayesian model is of significantly higher dimension as the state variables for the background traffic are continuous. Hence, solving the optimality equation $g + V = TV$ is notoriously hard, both analytically and numerically. In general, the Bayesian updates result in posterior distributions that cannot be captured by a nice structural form. In our problem, however, the decision maker can distinguish foreground and background upon ar-
rival leading to an arrival process with deterministic state transitions. It is only the 
departures that carry uncertainty with them. This leads to a state transition function, 
as described above, which keeps the dimensionality of the state space at reasonably 
low levels. In this way, the structure of the problem makes the Bayesian model a 
tractable approach (after discretization of the state space). Also note that for arbi-
trary nodes $i$ and $j$, the decision as to whether an incoming foreground job should 
join node $i$ or $j$, does not depend on the other nodes. Hence, in the decision making 
one can compare node 1 and 2, take the best node and compare it to node 3, take 
the best of that comparison and compare it to node 4, and so forth. This leads to a 
sequence of $N - 1$ comparisons. Therefore, the Bayesian approach scales linearly 
in running time with the number of nodes $N$.

4.3.2. Remark (Accuracy). In a general Bayesian setting, the belief state represents a 
probability distribution that represents the likelihood that the process is in a particu-
lar state. The accuracy of this estimate, generally, tends to deteriorate as the process 
progresses due to accumulated errors. In our problem setting, the accuracy of the 
estimates tends to improve as jobs leave the system. As more jobs leave the system, 
the support of the posterior distribution reduces to a smaller set of states, limiting 
the possibilities for errors. In fact, upon departure of the last job in a particular node, 
the posterior distribution of that node is independent of the past, since the state is 
extactly known. Thus, all probability mass is concentrated on having 0 jobs in that 
node. Hence, an empty node leads to a belief state that corresponds to the true 
state for that node. This observation increases the accuracy of our algorithm due to 
stability of the system.

4.3.2.2 Conditional sojourn time approach

In this section we propose a heuristic policy for near-optimal dispatching in the case 
of partial information, i.e. the dispatcher only has knowledge of the total numbers of 
(foreground plus background) jobs at each node. The policy is based on the combi-
nation of two policies that perform well on complementary sets of parameter com-
binations (see also the discussion below): (1) the Weighted Join the Shortest Queue 
(WJSQ) policy, and (2) the Conditional Sojourn Time (CST) policy.

The WJSQ policy routes an arriving foreground flow to the node where the total 
number of flows (normalized by the node speed) is minimal. Thus, the WJSQ for-
wards an incoming foreground job to node $i^*$, such that

$$
\gamma^{(\text{WJSQ})}_{j^*} = \min \left\{ \gamma^{(\text{WJSQ})}_1, \ldots, \gamma^{(\text{WJSQ})}_N \right\},
$$

(4.3)
Chapter 4: Efficient Traffic Splitting over Parallel Wireless Networks with Partial Information

where

\[ \gamma_i^{(\text{WJSQ})} := \frac{z_i}{C_i} \quad (i = 1, \ldots, N). \]

In other words, the WJSQ routes foreground flows to the node with the smallest \( z_i/C_i \) ratio. Ties are broken evenly. The WJSQ may be expected to work particularly well when the total load to the system is large (i.e., \( \rho \approx 1 \)) and the foreground load represents a significant fraction of the total load offered to the system (i.e., \( \beta \approx 1 \)).

The CST approach routes an incoming flow to the node for which the expected sojourn time, conditioned on the fact that there are \( z_i \) other flows at node \( i \) at that moment, is minimal. Using a well-known result for the conditional expected sojourn time in an M/M/1-PS queue [100], the CST policy forwards an incoming foreground job to node \( i^* \), such that

\[
\gamma_{i^*}^{(\text{CST})} = \min \left\{ \gamma_1^{(\text{CST})}, \ldots, \gamma_N^{(\text{CST})} \right\},
\]

(4.4)

where

\[ \gamma_i^{(\text{CST})} := \frac{z_i + 2}{2\mu C_i - \lambda_i} \quad (i = 1, \ldots, N). \]

The CST approach may be expected to work well if the foreground load is negligible compared to the total load (i.e., \( \beta \approx 0 \)). If the foreground load is large, then the dynamic decision making will induce a correlation between the number of flows in a PS node and the combined arrival process into that node, which leads to a violation of the Poisson assumption that underlies (4.4).

Both the WJSQ and the CST policies generate a switching curve given the total number of flows \( z_i \) on each node. We aim to develop a method that works well for the whole range of foreground and background load values. To this end, we propose a method where both switching curves are combined using a convex combination of these curves. The convex combination (CC) approach forwards an incoming foreground job to node \( i^* \), such that

\[
\gamma_{i^*}^{(\text{CC})} = \min \left\{ \gamma_1^{(\text{CC})}, \ldots, \gamma_N^{(\text{CC})} \right\},
\]

(4.5)

where

\[ \gamma_i^{(\text{CC})} := \alpha \frac{z_i}{C_i} + (1 - \alpha) \frac{z_i + 2}{2\mu C_i - \lambda_i}, \]

\[ (i = 1, \ldots, N). \]
and where $\alpha$ ($0 \leq \alpha \leq 1$) is given by:

$$
\alpha := \frac{\rho_0}{\sum_{i=1}^{N} C_i (1 - \rho_i)} \quad (i = 1, \ldots, N).
$$

Thus, the CST method is expected to work well when $\alpha \approx 0$, whereas the WJSQ method is expected to work well when $\alpha \approx 1$. In the next section the CC-approach defined in (4.5)-(4.6), and the performance of each of the policies discussed above is evaluated by simulations.

### 4.4 Numerical experiments

To assess the performance of the index rules discussed in Section 4.3 for efficiently assigning downloads with concurrent access based on partial state information, we have performed extensive experimentation with a state-of-the-art network simulation package OPNET [87], using an implementation for FTP file transfers via TCP/IP over two parallel WLANs. We have performed a large number of experiments with a wide range of parameter settings. The results are outlined below.

#### 4.4.1 Experimental configuration

In the experimental setup all wireless terminals download files from an application server, which may also be a dispatcher in front of several application servers (not shown). The application server has information about the number of ongoing downloads over each of the WLAN access networks, AP1 and AP2, but is unable to distinguish between the multi-homed and the single homed terminals, because there is no binding between both network addresses of the multi-homed terminals. Both WLAN access points operate on non-overlapping frequency channels to establish two non-interfering parallel paths to the application server from the multi-homed systems. The transmission links from the access points towards the application server are considered to incur a negligible delay and loss to packets from and to the access points. This assumption is motivated by the much higher capacities and reliability offered in contemporary fixed-line carrier-grade Internet connections in comparison to the IEEE 802.11b access networks. The analytic model from [59] captures the combined dynamics and protocol overhead of the 802.11 MAC, IP, TCP and application-layer into an explicit expression for the effective service time of a file download. Based on the effective service time, the effective load can be determined of the file transfers in our simulated WLAN networks with a flow-level $M/G/1$ PS model.
Figure 4.2: The experimental setup.

In the simulated network there are ten multi-homed terminals (named $FG_{01} - FG_{10}$) that generate download requests (that are considered foreground jobs in the queuing model) with arrival rate $\lambda_0$. These foreground terminals are positioned between both access points in a circle with a radius of 15 meter. In addition there are ten single-homed terminals (named with prefix $BG_{AP1}$) that generate background traffic in network 1 with file downloads arriving with rate $\lambda_1$ to the first network. The remaining ten single-homed terminals (named with prefix $BG_{AP2}$) generate background traffic at rate $\lambda_2$ in network 2 in a similar fashion. All background terminals are positioned at an equal distance of 15 meter from their respective access point. The file download requests arrive according to an independent Poisson process and may have multiple file transfers in progress.

The MAC/PHY parameters of the WLAN stations are set in accordance to the widely deployed IEEE 802.11b standard amendment as it relies on the same MAC protocol basis as the contemporary higher rate (IEEE 802.11 a/g/n) amendments and has lower computational requirements for high-load network simulations. Table 4.1 summarizes the IEEE 802.11 MAC parameters used in our analytic model to calculate the effective load values for the simulation runs. In this table, mac is the number
of bits of overhead bits associated to a MAC data frame. The difs, sifs, eifs are the DCF, short and extended interframe spacing times, respectively. The $\delta$ is the propagation delay that is assumed in our analytic model. $R_c$ is the transmission rate for WLAN acknowledgments of size ack bits, and $R_b$ is the WLAN transmission rate for MAC data frames that is set to 1 or 11 Mbps. $C_{W_{\text{min}}}$ corresponds to the minimum contention window in slots. Phy is the physical layer overhead, and $\tau$ is the slot time. In addition to the WLAN MAC, specific settings apply to the higher protocol layers and are outlined in Table 4.2. In Table 4.2, $X_{\text{FTPget}}$ is the size of the FTP GET-

<table>
<thead>
<tr>
<th>Variable</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{\text{FTPget}}$</td>
<td>4096 bits</td>
</tr>
<tr>
<td>$X_{\text{FTPclose}}$</td>
<td>64 bits</td>
</tr>
<tr>
<td>$\text{TCP}_{\text{stack}}$</td>
<td>Full-Featured</td>
</tr>
<tr>
<td>$X_{\text{MSS}}$</td>
<td>11584 bits</td>
</tr>
<tr>
<td>$X_{\text{tcp/ip}}$</td>
<td>416 bits</td>
</tr>
<tr>
<td>$w$</td>
<td>70080 bits (8760 bytes)</td>
</tr>
<tr>
<td>$X_{\text{file}}$</td>
<td>$1.6 \cdot 10^6$ bits</td>
</tr>
</tbody>
</table>

Table 4.2: Network and application settings.

command that is issued for initiating a file download, $X_{\text{FTPclose}}$ is the size of the FTP CLOSE-command that concludes the file transfer at the application. The TCP stack used in our experiments is characterized in OPNET as ‘Full-Featured’, which is an enhanced version of TCP Reno that uses Selective Acknowledgments (SACK) [82] and has a slightly smaller MSS, $X_{\text{MSS}}$ (in bits), due to the use of timestamps to fit in the 1500 bytes that are used as the WLAN data frame payload. The number of TCP/IP overhead bits per segment is $X_{\text{tcp/ip}}$ bits. The maximum TCP receiver window size is indicated as $w$ (in bits), and the file size as $X_{\text{file}}$ (in bits). Based on the parameter setting from Table 4.1 and 4.2 and respecting the engineering guidelines from [59] we can assume that the mean download response times in our simulation model can be accurately predicted from the effective load of the network using the $M/G/1$ PS model.
4.4.2 Experimental results

The OPNET simulations for the experimental results have been run with approximately 322,000 foreground jobs and the background jobs ranging from roughly 644,000 jobs to 5.1 million jobs depending on the load. In our simulation study we have considered two scenarios. One simulation scenario considers equal capacity networks in which all terminals are configured to use a WLAN transmission rate of 11 Mbps. For simulating a scenario in which the network capacity of both access network is unequal, the WLAN transmission rate used in AP2 is lowered to 1 Mbps, which reduces the medium capacity for processing file transfers by a factor of 5.79. In this scenario, the background load applied to AP2 is based on the lower capacity, whereas the foreground traffic intensity remains the same as for the equal capacity network. We have executed 48 runs for the equal capacity scenario (24 runs for the fully observed MDP and 24 runs for the heuristics) and 80 runs for the unequal capacity scenario. All runs have completed a total simulation time of 300 hours per run of which 1 hour is the warm-up time leading to a wall clock time of approximately 75 hours per run. This experimental setup is sufficient to derive a 99% confidence interval of approximately 0.7% with respect to the point estimates.

To assess the efficiency of the different partial-information policies, we have simulated the mean transfer time of an arbitrary foreground job, $E[S_0]$, for different policies, and compare the outcome to the full MDP case. For given policy $\pi$, the relative error is defined as follows:

$$
\Delta\% = \frac{E[S_0|\pi] - E[S_0|\text{full MDP}]}{E[S_0|\text{full MDP}]} \times 100%. 
$$

(4.7)

Note that the simulations have been run with $10^7$ foreground jobs resulting in a 99% confidence interval of approximately 0.1% with respect to the point estimates.

4.4.2.1 The case of equal capacities

We first consider the case where both access networks have the same (normalized) capacity, i.e., $C_1 = C_2$. The results of the experiments are outlined in Tables 4.3 to 4.5, for $\rho_0 = 0.1$ and a number of combinations $\rho_1$ and $\rho_2$. Tables 4.3 and 4.4 show the results for the (W)JSQ and the CC policies, benchmarked against the full MDP policy. Table 4.5 shows a comparison between the CC policy and the Bayesian policy [14]. Note that the parameter values are obtained according to the parameterization as defined and validated in [59]. The results in Table 4.3 show that the JSQ policy performs quite well, with a maximum error up to 4.7%. However, the results in Table 4.4 show that the CC policy strongly outperforms JSQ, with a maximum error
of 0.9%. The difference in performance between JSQ and CC manifests itself mainly when the background load values are strongly asymmetric. In those cases the JSQ policy becomes highly inaccurate. To illustrate this, consider a two-node system where node 1 has high background load and node 2 has low background load. If $n_1 < n_2$ then the JSQ policy will route an incoming job $T$ to node 1. In this situation, it may well occur that this decision is not optimal, because the sojourn time of $T$ is likely to be stretched due to the background job arrivals at node 1. Table 4.5 shows that the CC policy performs comparably well to the Bayesian policy, despite the fact that the Bayesian policy has a much higher computational complexity. We re-emphasize that the computational complexity of the CC rule is negligible.
4.4.2.2 The case of unequal capacities

Let us now consider the case where the access networks have different capacities, i.e., \( C_1 \neq C_2 \). To this end, we consider the case \( C_1 : C_2 = 1 : 0.17 \). The results of the simulations experiments are outlined in Tables 4.6 to 4.8, for \( \rho_0 = 0.1 \) and a number of combinations \( \rho_1 \) and \( \rho_2 \). Tables 4.6 and 4.7 show the results for the WJSQ and the CC policies, benchmarked against the full MDP policy (similar to the results in Tables 4.3 and 4.4 for the equal capacity case). Table 4.8 shows a comparison between the CC policy and the Bayesian policy (similar to Table 4.5, see Section 4.2.1). Recall that the parameter values in this setting are obtained according to the parameterization discussed in [59].

![Table 4.6: Comparison of \( \mathbb{E}[S_0|\text{WJSQ}] \) and \( \mathbb{E}[S_0|\text{full MDP}] \) for \( \rho_0 = 0.1 \) with \( C_1 = 1 \) and \( C_2 = 0.17 \).](image)

<table>
<thead>
<tr>
<th>( \rho_1 )</th>
<th>( \rho_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>(1.827, 0.415, 147.5%)</td>
</tr>
<tr>
<td>0.2</td>
<td>(1.094, 0.474, 130.7%)</td>
</tr>
<tr>
<td>0.3</td>
<td>(1.741, 0.555, 213.8%)</td>
</tr>
<tr>
<td>0.4</td>
<td>(1.247, 0.660, 88.9%)</td>
</tr>
<tr>
<td>0.5</td>
<td>(1.348, 0.806, 67.1%)</td>
</tr>
<tr>
<td>0.6</td>
<td>(1.498, 1.013, 47.8%)</td>
</tr>
<tr>
<td>0.7</td>
<td>(1.714, 1.305, 31.3%)</td>
</tr>
<tr>
<td>0.8</td>
<td>(2.093, 1.777, 17.8%)</td>
</tr>
</tbody>
</table>

![Table 4.7: Comparison of \( \mathbb{E}[S_0|\text{CC}] \) and \( \mathbb{E}[S_0|\text{full MDP}] \) for \( \rho_0 = 0.1 \) with \( C_1 = 1 \) and \( C_2 = 0.17 \).](image)

<table>
<thead>
<tr>
<th>( \rho_1 )</th>
<th>( \rho_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>(0.418, 0.415, 0.8%)</td>
</tr>
<tr>
<td>0.2</td>
<td>(0.476, 0.474, 0.3%)</td>
</tr>
<tr>
<td>0.3</td>
<td>(0.556, 0.555, 0.3%)</td>
</tr>
<tr>
<td>0.4</td>
<td>(0.664, 0.660, 0.6%)</td>
</tr>
<tr>
<td>0.5</td>
<td>(0.809, 0.806, 0.3%)</td>
</tr>
<tr>
<td>0.6</td>
<td>(1.019, 1.013, 0.6%)</td>
</tr>
<tr>
<td>0.7</td>
<td>(1.331, 1.305, 2.0%)</td>
</tr>
<tr>
<td>0.8</td>
<td>(1.841, 1.777, 3.6%)</td>
</tr>
</tbody>
</table>

The results in Table 4.6 show that the WJSQ policy is highly inefficient when the network capacities are strongly asymmetric, with error even up to over 200%. The results reveal that the WJSQ performs particularly bad in low-load scenarios. However, the results in Table 4.7 show that the CC policy remains to be highly efficient, even when the networks are strongly asymmetric, with a worst-case error less than 4%. Table 4.8 shows again that the CC policy performs comparably well to the Bayesian policy, and that both policies are highly accurate.
4.5 Discussion

<table>
<thead>
<tr>
<th></th>
<th>$0.1$</th>
<th>$0.3$</th>
<th>$0.5$</th>
<th>$0.7$</th>
<th>$0.8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.1$</td>
<td>(0.418, 0.417, 0.4%)</td>
<td>(0.415, 0.415, 0.0%)</td>
<td>(0.416, 0.416, 0.2%)</td>
<td>(0.416, 0.416, 0.2%)</td>
<td>(0.415, 0.415, 0.0%)</td>
</tr>
<tr>
<td>$0.2$</td>
<td>(0.476, 0.476, −0.1%)</td>
<td>(0.476, 0.477, −0.2%)</td>
<td>(0.477, 0.479, −0.4%)</td>
<td>(0.477, 0.479, −0.4%)</td>
<td>(0.475, 0.475, 0.0%)</td>
</tr>
<tr>
<td>$0.3$</td>
<td>(0.556, 0.556, 0.1%)</td>
<td>(0.556, 0.556, −0.1%)</td>
<td>(0.555, 0.553, 0.4%)</td>
<td>(0.555, 0.553, 0.4%)</td>
<td>(0.555, 0.555, 0.0%)</td>
</tr>
<tr>
<td>$0.4$</td>
<td>(0.664, 0.663, 0.2%)</td>
<td>(0.666, 0.665, 0.1%)</td>
<td>(0.667, 0.667, 0.0%)</td>
<td>(0.667, 0.667, 0.0%)</td>
<td>(0.667, 0.667, 0.0%)</td>
</tr>
<tr>
<td>$0.5$</td>
<td>(0.809, 0.807, 0.3%)</td>
<td>(0.821, 0.819, 0.2%)</td>
<td>(0.831, 0.827, 0.5%)</td>
<td>(0.831, 0.827, 0.5%)</td>
<td>(0.834, 0.835, −0.2%)</td>
</tr>
<tr>
<td>$0.6$</td>
<td>(1.019, 1.016, 0.3%)</td>
<td>(1.047, 1.052, −0.4%)</td>
<td>(1.070, 1.068, 0.2%)</td>
<td>(1.070, 1.068, 0.2%)</td>
<td>(1.100, 1.100, 0.1%)</td>
</tr>
<tr>
<td>$0.7$</td>
<td>(1.331, 1.322, 0.7%)</td>
<td>(1.399, 1.410, −0.8%)</td>
<td>(1.477, 1.482, −0.4%)</td>
<td>(1.477, 1.482, −0.4%)</td>
<td>(1.602, 1.597, 0.3%)</td>
</tr>
<tr>
<td>$0.8$</td>
<td>(1.841, 1.817, 1.3%)</td>
<td>(2.023, 2.057, −1.0%)</td>
<td>(2.313, 2.299, 0.6%)</td>
<td>(2.313, 2.299, 0.6%)</td>
<td>(2.866, 2.876, −0.4%)</td>
</tr>
</tbody>
</table>

Table 4.8: Comparison of $E[S_0|\text{Bayes}]$ and $E[S_0|\text{CC}]$ for $\rho_0 = 0.1$ with $C_1 = 1$ and $C_2 = 0.17$.

4.4.2 Varying the asymmetry in foreground versus background load

Finally, we check the efficiency of the splitting policies where we vary the fraction of the foreground compared to the total load. Similar to Section 4.2.2, we assume that the ratio of the network capacities are $C_1 : C_2 = 1 : 0.17$. Figures 4.3a to 4.3c show the expected value of the transfer time of an arbitrary foreground job (i.e., $E[S_0]$) as a function of the ratio $\beta = \rho_0/\rho$, where the overall load $\rho$ is kept fixed, for each of the routing policies CC, WJSQ, full MDP and Bayes. Figures 4.3a, 4.3b and 4.3c show the results for $\rho = 0.65$, $\rho = 0.70$ and $\rho = 0.80$, respectively.

The results in Figures 4.3a to 4.3c show again that in all cases the WJSQ policy is strongly outperformed by the other policies. Moreover, we observe that the CC policy, which is based on partial information only, is extremely close to the full MDP solution, which is based on full state information. Also, we observe our simplistic index-based CC rule performs comparably well to the more complicated Bayesian policy. We re-emphasize that the importance of this observation for practical engineering purposes.

4.5 Discussion

In conclusion, the experimental results demonstrate that the CC-method using partial information strongly outperforms the WJSQ policy, and even leads to close-to-optimal performance that can be obtained using the full-state information MDP. Moreover, the CC method performs equally well when compared to the Bayesian policy, which has a number of drawbacks: (1) it is inherently complicated, which limits its practical usefulness, (2) it is not scalable in the number of access networks $N$, because it needs the full-state information MDP solution, which suffers from the curse of dimensionality. Typically, this will limit the applicability of the Bayesian methods due to memory constraints. These observations lead to the conclusion
that the CC index rule has a considerable advantage over the Bayesian approach with respect to its practical usefulness and engineering.

Figure 4.3: Comparison of $E[S_0|CC]$, $E[S_0|full\ MDP]$, $E[S|WJSQ]$ and $E[S_0|Bayes]$ in OPNET for different $\rho$. 