Acknowledgment

I would like to thank Dr. Nasser Yazdani, my advisor in this thesis. Also, I would like to say a special thank you to Mr. Sajjad Zarifzadeh for his generous helps and advices throughout the thesis.

1 Contents

1 Introduction

According to IP networks definition, along the path of a flow from the source to the destination, any node (router) including the source can change the forwarding path of the packets (Figure 1-1). One step further, the node can intelligently split the received packets among available forwarding paths. This can balance the load on the network routers and hence i) avoids congestions in bottleneck nodes and ii) makes efficient use of underutilized routers for which considerable amount of money have already been invested. This technique is called *multipath routing*.

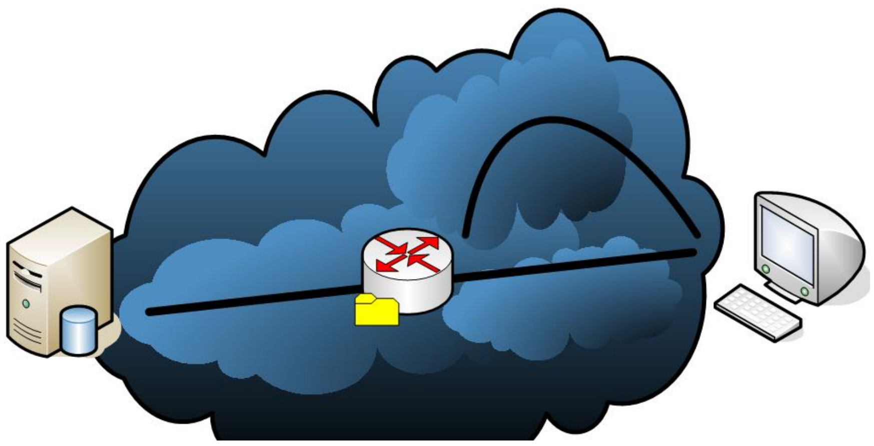


Figure 1-1: The possibility of routing through multiple paths in IP networks.

The special case of multipath routing in which the source is the split point between the packets is known as *multipath transferring*. If the source transfers the packets of the same flow through multiple paths, then it is called *concurrent multipath transferring*. A host can apply multipath transferring if i) it is connected to the Internet through multiple network interfaces or ii) it uses multiple gateways to access the Internet.

The rapid development of IP networks and specially the widespread deployment of wireless networks has provided the users with the option of choosing among several available interfaces for accessing the Internet (Figure 1-2). For example, a user might have access to the Internet through different technologies such as Ethernet, 802.11, Bluetooth, 3G, and PSTN Modem, such that each technology has its own Internet service provider (ISP). Selecting one of the available services for accessing the

Internet is a dynamic optimization problem which should consider variety of metrics including network bandwidth, end-to-end delay, delay variance, Quality of Service (QoS), cost, power consumption, signal interference, battery lifetime, and traffic patterns.

Some organizations provide their users with several gateways to access the Internet. Even with one interface card, each user can take advantage of multipath transferring by splitting its traffic among the available gateways.

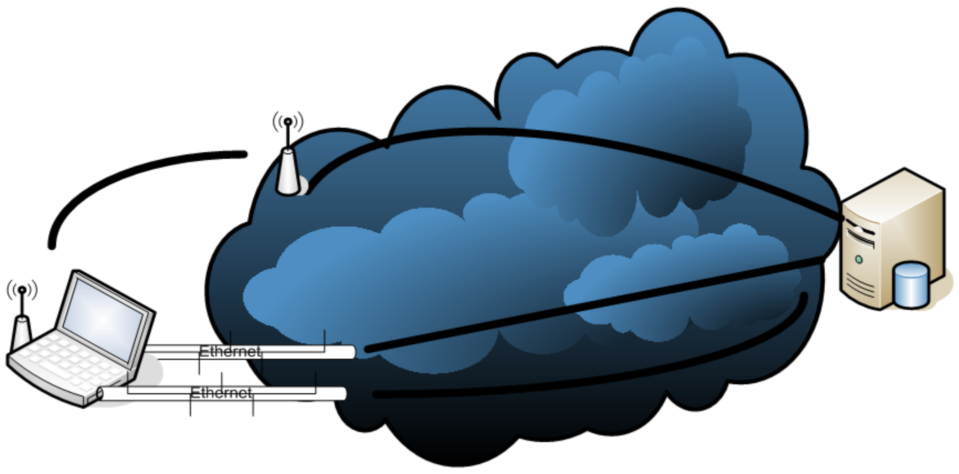


Figure 1-2: The capability of nowadays hosts to support concurrent multipath transferring

Bandwidth is usually a scarce resource especially in wireless networks. Hence, the users of mobile nodes or cell phones might prefer to use multiple network interfaces concurrently. Another motivation for using more than one network interface is error resilience; if one of the network interfaces fails or router congestion happens along its corresponding path, then the connection will still carry on via the other paths.

By concurrently transferring the packets of a same flow over multiple paths, the packets which take different paths will experience different delays. As a consequence, they will be delivered to the destination reordered. The *reordering* causes different problems for different transport protocols. The goal of this thesis is to address these issues by making only small modifications at the IP layer.

1.1 Benefits of multipath transferring

We briefly mentioned some benefits of transferring through multiple paths. In this section, we explain the benefits of multipath transferring in detail.

1.1.1 Load balancing

Congestion is one of the top open problems of computer networks. When the rate of incoming packets exceeds the outbound capacity, the router keeps the incoming packets into a buffer. This addresses the momentary increase in the load. However, if the rate of incoming packets stays high, the buffer finally overloads and the router has to drop some packets. The router is then called congested. At the same time that some bottleneck routers are congested, there might be lots of routers spread over the Internet that are working way below their optimum capacity.

Load balancing is one way to address the congestion problem. The goal is to move part of traffic from the overloaded routers to the other idle ones. Load balancing on one hand decreases the probability of congestion in bottleneck nodes and on the other hand makes an efficient use of other underutilized routers whose deployment has already cost us.

In the context of wireless ad hoc networks, unbalanced traffic makes the bottleneck nodes to run out of battery sooner than the others. By balancing the load, we can make sure that they will be serving for longer times.

Multipath transferring (and multipath routing in general) is one of the proposed approaches for load balancing. By splitting the traffic over multiple paths, there will be less traffic carried on each path and thus the load will be more balanced on the intermediate routers.

1.1.2 More aggregate bandwidth

The today’s end hosts have access to the Internet through different technologies such as Ethernet, 802.11, Bluetooth, 3G, and PSTN Modem, such that each technology has possibly its own ISP. The application throughput is either limited by the interface bandwidth or at a gateway by some Internet service providers (it can be the ISP to which we directly connect or a higher level ISP). By concurrently using multiple interfaces or gateways, we can achieve an aggregate throughput which is more than the available bandwidth through each of the interfaces or gateways individually.

In wireless ad hoc networks, bandwidth is a scarce resource and the intermediate nodes can split the outgoing traffic among multiple forwarders to alleviate the limited bandwidth problem.

1.1.3 More reliability or error-resilience

Several reasons can be the cause of a link failure including the failure of the node to which the link is connected, cable cut (after a shark attacks undersea cables for example), or congestion at the overloaded router that the link ends into. Having an alternate route ready to use, the nodes can pass over the failed link by transmitting the packets over another routes. The same argument holds for the multipath transferring: a connection does not break after a path failure if the end host has been using multiple paths for the connection.

Furthermore, the quality of service (QoS) provided by different paths is proportional to their cost; the paths which provide QoS are more expensive and the paths that work based on best-effort policy are cheaper. To optimize the cost, we can split the traffic over multiple paths in a way that the most important data go though the high quality path and the rest of traffic which are less important are transmitted via cheap, best-effort paths. Thus, the connection does not break even after an error in the best-effort path, although the overall quality decreases. For example, in multimedia applications, part of traffic is vital to have a basic communication and the other parts only increase the quality of the received stream. Some multimedia protocols such as H.263 support two separate output streams for this purpose.

1.2 Challenges in concurrent multipath transferring

There are certain challenges in concurrently transferring the packets of the same flow over multiple paths. Plainly, such a scheme causes the packets which are transmitted over different paths, to experience different delays. In wired networks the selected paths can have different delays because of i) different traffic loaded on intermediate routers, ii) different path lengths in terms of number of intermediate routers, and iii) different network media such as satellite vs. optimal links.

Wireless networks have to deal with another problem that is the potential interference between paths which leads to more packet loss or longer transmission delays. To overcome this deficiency, the selected paths should be completely disjoint and also geographically separated, as far as possible. Utilizing more diverse paths implies more discrepancy between their delays. On the other hand, recent study (Ganjali & Keshavarzian, 2004) shows that load balancing cannot be really achieved in multipath schemes, unless numerous paths are used to convey the packets. However, using more paths entails more variations among their delays.

In the following, we discuss the problems caused by different paths delays as well as specifically packet reordering.

1.2.1 TCP timeout

TCP offers reliable transmission in the sense that it detects packet loss through the received acknowledgements (ACKs) from the destination and retransmit them. If the received ACKs do not indicate the receipt of a packet, after passing RTO (Retransmission Timeout) unit of time, TCP deduces that the packet is lost and thus retransmits the packet. Besides, to avoid any possible congestion, it quickly backs off by reducing the congestion window to zero, i.e., the throughput drops to zero. The process is depicted in Figure 1-3.

One problem of TCP under concurrent multipath transferring is that the RTT (Round Trip Time) estimation is not accurate; different delays over different paths lead to unstable RTT estimation and incorrect RTO setting. For example, if the RTT difference of the longest path and the shortest path is large, TCP could prematurely timeout packets on the longest path due to the incorrect RTT estimation.

The unnecessary timeout of the packets has two major negative impacts on TCP:

1. The unnecessary retransmission of the packets which wastes the network bandwidth.

2. In addition, after each timeout, TCP drastically scales down its congestion window size to zero and invokes the slow-start mechanism, and causes underutilizing the paths and further degrading the performance of multipath scheme.

In this thesis, we specify the conditions that holding them will avoid the unnecessary timeout in TCP.

1.2.2 Reordering

The different path delays yield the packets to be delivered reordered. The more difference between end-to-end path delays, the more packets are delivered out-of-order. This causes problems both in TCP and UDP which we explain in the following.

1.2.2.1 Unnecessary trigger of fast-retransmit/recovery in TCP

ACKs in TCP are accumulative, meaning that the ACK for sequence number x indicates that all the bytes with less sequence number are received and the receiver is now expecting sequence number x (recall that TCP is stream-oriented protocol, hence the ACKs are for byte and not packets). A duplicate ACK for sequence number x can implies that the packet which included byte x is not received yet, perhaps reordered. Receiving more duplicate ACKs is good signal that probably the packet is lost. Most TCP implementations include the ‘‘fast-retransmit/recovery’’ mechanism (Jacobson, April 1990) which is illustrated in Figure 1-3. In short, fast-retransmit/recovery mechanism by receipt of the *dupthresh* duplicate ACK (which is mostly set to three), assumes that the packet is lost. It then retransmits the packet and also reduces the congestion window size by half. The latter is to avoid probable congestion in an intermediate router.

In concurrent multipath transferring, even if the RTO is set high enough to prevent the false timeouts, fast-retransmit/recovery may still incur another serious problem. Here, we bring a simple example to illuminate this issue. Suppose the ratio of the RTTs of two employed paths is four. Furthermore, assume that the first packet is transmitted on the slower path and the next four packets are transmitted over the faster path. The receipt of packets 2, 3, and 4 at the destination will cause the receiver-side TCP to signal a fast-retransmit/recovery of the first packet, hence wasting the prior transmission on the slower path. Worse yet, the recovery phase following a fast-retransmit/recovery scales down the congestion window to the half of its current size, resulting in a high degradation in the throughput of the connections.

Recent studies have shown that false fast-retransmit/recovery can be reduced by increasing the triggering threshold, *dupthresh*, according to the underlying network conditions (Blanton, et al., 2002)(Zhang, et al., 2003). However, the approach in (Zhang, et al., 2003) imposes excessive computational and storage overheads to construct a reordering histogram, whereas the techniques proposed in (Blanton, et al., 2002) may not well adapt to changing network conditions promptly. Moreover, (Ma, et al., 2004) has studied this approach under multipath network scenarios while it does not investigate the performance of the proposed method in networks with diverse delays and bandwidths.

All these approaches have the drawback of interfering with normal operation of TCP. Moreover, like other proposals which need modifications in the transport layer, the wide-spread deployment of the solution would encounter serious challenges and thus is unlikely to happen. We will discuss it more in the next section when we explain the category of the solutions which operate on the transport layer. In this thesis, we propose a novel approach at the IP layer to alleviate this problem.

1.2.2.2 Need for larger buffers at the receiver side

As multipath transferring intensifies the reordering problem, the application layer demands larger buffers to keep out of order received packets. This raises difficulties for usage of multipath transferring scheme in receiver devices with limited resources, such as handled PDA and sensor.

Furthermore, since the receiver has to wait for the packets sent through the slower path to be received, the overall observed delay by the application is the delay of the slower path. In other words, no matter how fast is the other path, the receiver application can not benefit from it.

Even though UDP based protocols such as RTP has been extended to support multipath schema (Mao, et al., 2003), but they do not propose solutions for the reordering caused by the multipath transferring.

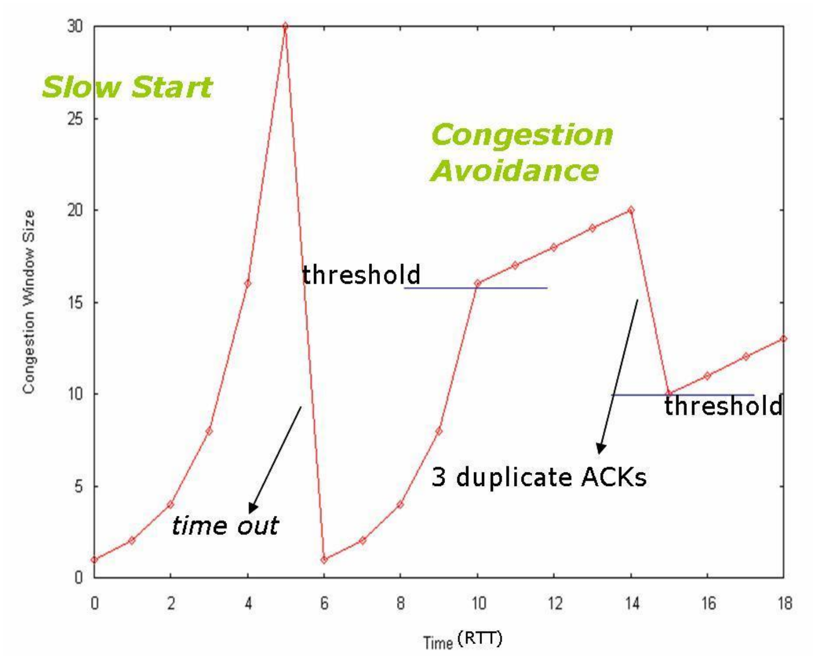


Figure 1-3: The changes in TCP congestion window size.[1]

1.3 Thesis structure

In this thesis, we propose two approaches at the IP layer to address the reordering problem of TCP and UDP. In the case of TCP, the key observation is that the interleaved reception of the packets sent via multiple paths, at the destination does not trigger the fast-retransmit/recovery timer, even though the packets are received reordered. Therefore, the IP layer who is in charge of alternating the packets among the multipath available paths needs to linger on the slower path for at least the delay difference between the paths. Moreover, we introduce an analytical model to estimate the probability of triggering fast-retransmit/recovery by our method. In the case of

[1] The picture is borrowed from http://www.cs.nccu.edu.tw/~lien/TALK/SAHNS07/hardcopy.htm

UDP, we first analyze the reordering problem through a sociological perspective. Under the light of the presented model, we propose to schedule the packets at the source to have them received in-order at the destination. We also present an analytical model to compute the buffer size at the destination.

In Section 2, we categorize the possible strategies for multipath transferring (and in general multipath routing). At the end of the section, we demonstrate the position of the proposed approaches in this thesis among the explained categories. This section also covers examples of some most important related works which have tackled the reordering problem in concurrent multipath transferring. In Section 3, we introduce novel approaches operating at the IP layer, to address the reordering problem of concurrent multipath transferring schemes in TCP and UDP. We also develop an analytical model to demonstrate the efficiency of the proposed approaches. Section 4 evaluates the performance of the proposed approaches through simulation results. We conclude the thesis in Section 5 and discuss the future works. The reported results in the thesis have been published in two journal papers (Yabandeh, et al., March 2007) (Yabandeh, et al., 2007) and one conference paper (Yabandeh, et al., 2008).

2 Multipath routing strategies

In this section, we categorize the different strategies for multipath routing. At the end of this section, we identify the position of the proposed approaches in this thesis among the explained categories.

2.1 Operating phase

Multipath routing consists of two basic phases: i) creating a set of paths and making them available and ii) transferring data via the created paths. These two phases are separate and each of them has its own design issues(Yabandeh, et al., 2007). Each multipath routing algorithm might operate over one of the phases or both of them.

In this thesis, we assume that the paths have been already created and are ready to use. The proposed algorithms then focus on transferring the data through the available multiple paths. In the following, we briefly explain each phase.

2.1.1 Creating multiple paths

As explained above, the first phase of multipath routing is to create the multiple paths. Creating the paths can be as simple as accessing multiple network interfaces which are already available on the machine, or as difficult as creating multiple node-disjoint paths in wireless ad hoc networks. Some organizations provide the users with multiple gateways. The IP layer then can split the outgoing traffic between the gateways.

In the general multipath routing where a intermediate router is in charge of creating multiple paths, the router can split the incoming traffic between multiple next hub (router).

2.1.2 Transferring through multiple paths

There are different strategies in transferring data through multiple paths. If the only aim of multipath routing is achieving error-resilience, one possible strategy could be redundant transferring of data through all of the paths. In this way, by a failure in one path, the data is still delivered uninterruptedly via other paths.

In other applications such as load balancing and achieving more aggregate bandwidth, the transmitted traffic through paths are disjoint. However, the sender can have several strategies for splitting the traffic among multiple paths. On one hand, it can split the flows among the paths and transfer all the packets of the same flow through a single path, and on the other hand it can split packets of each flow among the available paths. Making this decision, in fact specifies the *granularity* of the strategy. In this thesis, we adopted the latter approach: per-packet granularity. In the following, we discuss the pros. and cons. of each approach.

2.1.2.1 Per-flow granularity

The advantage of per-flow granularity is that the packets of the same flow do not experience diverse delays and thus the problems of different path delays and reordering are avoided automatically. Thus far, most of the multipath schemes have tried to bypass the reordering problem by taking the per-flow granularity (Lee, et al., 2001)(Wei, et al., 2004)(Marina, et al., 2001)(Ye, et al., 2003). Nevertheless, this approach is not resilient against path failures as it would take some time to switch to another path after a failure. This delay is not acceptable in real-time applications such as video conferencing.

2.1.2.2 Per-packet granularity

In per-packet granularity, the packets of the same flow can be split among multiple paths and hence experience different delays. TCP's timeout mechanism is designed only for one path and will be confused when the received ACKs report variant delays.

Also discrepancy between path delays implies the out-of-order delivery of the packets at the receiver. TCP's fast-retransmit/recovery mechanism tolerates up to 3 reordered packets as the reordering is a rare event under the normal single path routing. Beyond that, TCP assumes the packet as lost. However, in the multipath scheme the packets can be subject to much more reordering. It is worth noting that naively increasing the threshold would make the algorithm vulnerable to the actual packet lost events. Also in UDP, the more reordering demands more buffer space at the receiver side which is sometimes a scarce resource, such as in hand-held devices.

Nevertheless, as reported in (Krishnan, et al., 1993), a per-packet granularity results in much better performance than the explained per-flow streaming approach. This is because the packets can be evenly split among paths. This is not possible in per-flow granularity when there is a major flow (such as online streaming of a high-quality movie) that dedicates the most part of the outgoing traffic to itself.

In this thesis, we propose some approaches to alleviate the problems of per-packet granularity and allow the applications to take advantage of its benefits.

2.2 Deployment environment

Multipath routing has wide-spread usage both in wired and wireless networks. Considering the different requirements and conditions of these two environments, the proposed algorithms for these two environments, although established upon common notions, apply different strategies. In this thesis, we do not make assumptions regarding the environment and the proposed algorithms can be applied to both wired and wireless environments. In the following, we discuss the applied strategies which are dedicated to each of these two environments.

2.2.1 Wired networks

Historically in wired networks, multipath routing was initially applied to intermediate routers to dynamically split the incoming traffic among the available routes. It mainly aimed the following objectives: i) load balancing and ii) insuring quality of service by increasing the reliability.

Nowadays, having several network interfaces available for end users, the Internet users can take advantage of this opportunity and split the outgoing traffic among these interfaces. Some organizations provide their users with multiple choice for the gateway and the user can also split the outgoing traffic among the multiple gateways.

2.2.2 Wireless networks

By emergence of wireless ad hoc networks, new potentials for applying multipath routing were created. This is because most of the routing protocols in wireless ad hoc networks (such as DSR(Johnson, et al., 1999), AODV(Perkins, et al., 2000), ZRP(Haas, et al., 1999)) have the inherent capability to return multiple path upon a routing request. By emphasis on this potential, the new designed routing protocols extended the on-demand routing protocols to support multiple paths. For example SMR(Lee, et al., 2001), RMPSR(Wei, et al., 2004), AOMDV(Marina, et al., 2001), and AODVM(Ye, et al., 2003), are the multipath extended versions of DSR and AODV. Therefore, multipath routing was one of the candidates that can be deployed in wireless networks and alleviate its inherent problems, i.e., limited bandwidth and high error rate.

In summary, the proposed protocol based on multipath routing tried to address the following issues:

1. Achieve more aggregate bandwidth which is a scarce resource in wireless networks(Maxemchuc, 1975).
2. Balance the load over the nodes which can both increase the battery lifetime as well as avoid congestion in bottleneck nodes(Wu, et al., 1992).
3. Enhance the error-resilience which is the inherent problem in wireless networks due to noise and interference (Stewart, et al., 2001).

These issues become doubly important when it comes to real-time streaming of multimedia content which needs high bandwidth as well as higher level of error-resilience.

2.3 Split point

In multipath routing, the split of the packets can be done at any node along the path from the source to the destination. In the special case, the split point can be even the source of the flow. When the split point is an intermediate router, the applied strategy is mostly known as *multipath routing* and when the source is in charge of splitting the packets among multiple paths, it is usually called *multipath transferring*. We adopt the same terminology for the rest of the thesis.

The advantage of the multipath transferring scheme is that paths are selected by the source. In other words, the paths are selected in an end-to-end way. This strategy benefits from all the advantages of end-to-end approaches which fall into so-called end-to-end argument (Saltzer, et al., 1984). As a consequence, the complexity of the underlying network decreases and the proposed approaches will be more extendable.

Concurrent multipath transferring, which is the domain of this thesis, is the special case of multipath transferring which uses per-packet granularity.

2.4 Implementation layer

The multipath transferring technique can be implemented at any given layer of the networking stack: application, transport, IP, and physical layer. The implementation in the physical layer is for very particular applications and hence is not covered in this section. In the following, we categorize the related works based on the implementation layer and discuss the pros and cons of each category. The proposed algorithms in this thesis will be implemented at the IP layer.

2.4.1 Application layer

Some of the related works (Hacker, et al., 2002)(Sivakumar, et al., 2000) propose to use several independent TCP connection to increase the overall throughput in high speed networks. These applications in fact are using the same physical path for all the created TCP connections. This is different from concurrent multipath transferring where the data is split over multiple physical paths[2].

Content networks (Day, et al., 2003) provide an infrastructure for load balancing of TCP connection by using a per-flow granularity perspective. This is different from the focus of this thesis which is the per-packet granularity. Even though per-flow granularity is still useful for quick http requests/responses, still for the applications

[2] Note that in general there is still no guaranty that physical paths do not share some links or nodes.

which transfer a large bulk of data this schema is not optimal, because of using only one physical path for the flows.

To the best of our knowledge, there is no algorithm at the application layer which uses per-packet granularity. Even in such a case, because of many interactions between the application and the transport layer, it would be more reasonable to push such an algorithm to the transport layer. This is because concurrent multipath transferring algorithms require information regarding packet delay and loss to select the optimal path and this information are not available at the application layer. In general, implementing the concurrent multipath transferring at the application layer would increase the redundant code (one separate piece of code for each application) and consequently increase the chance of introducing programming bugs because of separate implementations.

2.4.2 Transport layer

The advantage of the protocols which work on the transport layer is that they have access to the information which is necessary to select the optimal path, such as delay and loss of packets. On the down side, the proposed solution would be specific to a particular transport protocol and others have to develop their own version of the multipath transferring, perhaps redundantly. Furthermore, some transport layer protocols such as TCP are very complicated as well as widely used in a variety of environments. Hence, every small change in the protocol needs thorough analysis and evaluation to prove that the performance of the protocol is not hurt in other usage scenarios. For example, a solution might improve the TCP performance in multipath applications but damage its performance in single path scenarios. Due to widespread usage of these protocols, the process of standardization and global deployment would be very difficult and often unlikely to happen.

In the following we name some of the proposed solutions in transport layer.

2.4.2.1 SCTP

Similar to TCP, SCTP (Stream Control Transmission Protocol) guaranties in-order delivery of packets(Stewart, et al., 2000). However, in contrast to TCP which is stream-oriented, SCTP is record-oriented in the sense that unit of transport is a message rather than a stream of bytes. Because of this fundamental difference, SCTP can operate on multiple streams (possibly with different IP addresses or network interfaces) at the same time and in-order delivery will be hold for each stream separately. Therefore, the reordering between messages of multiple streams does not make sense anymore and thus is not an issue.

However, the application must be aware of the availability of several streams and treat them differently. For example, web page images can be sent separately from web page texts. If the application has a single big coherent chuck of data to transfer, SCTP will not be able to split it over multiple streams.

So far, SCTP has used the secondary path either for retransmission traffic or redundant transmission of the primary traffic. By the latter usage, the stream of data will be continued uninterruptedly even in the case of primary path failure and consequently it provides more reliability. Later several works proposed solutions on top of SCTP for bandwidth aggregation (Abd El Al, et al., 2004)(Abd El Al, et al., 2004)(Argyriou, et al., 2003). In below, we explain one of them in more detail.

2.4.2.2 Bandwidth aggregation with SCTP

In (Argyriou, et al., 2003), it is proposed to transfer data concurrently over SCTP to achieve more aggregate bandwidth. It suggests changing the fast-retransmit/recovery technique in SCTP. However these changes assume presence of data which is not available at receiver side. For example, the assumption that the receiver can distinguish between a lost packet and a reordered packet is not realistic.

2.4.2.3 mTCP

mTCP (Zhang, et al., 2004) is an extension to TCP to achieve more aggregate bandwidth by concurrently transferring data through multipath paths. It uses mechanisms to detect a shared bottleneck (congestion point) among the paths and suppress all the paths with the same congestion. It is dependent on an overlay network named by RON (Andersen, et al., 2001) to create the multiple end-to-end paths and hence is not as general as TCP. It uses a single path for the ACK traffic and hence it is vulnerable to failure of the response path unless additional mechanisms are applied.

2.4.2.4 pTCP

Parallel TCP stripes the traffic of wireless ad hoc networks over multiple virtual TCP paths, TCP-v, where each TCP-v is a separate TCP connection. Each TCP-v is in charge of congestion control and retransmission over its path. A separate manager unit decides which data must be transferred with which TCP-v. Separating the functionalities enables the pTCP to intelligently schedule the transmission and retransmission of data.

pTCP is limited to only wireless ad hoc networks and its operation over wired networks is not investigated. On the other hand, it suffers from complicated design and implementation.

2.4.3 IP layer

The advantage of implementing multipath transferring at the IP layer is that it has access to all information regarding the existence and number of multiple paths. Multiple paths might be available through multiple network interface card or multiple gateways that in both cases the IP layer has full information about them. Moreover, all the devices which implement the IP layer, including intermediate routers, can take advantage of the proposed solutions in this layer. This in oppose of transport layer solutions which is only deployable at the end hosts.

The disadvantage is that sometimes choosing the optimal path requires some information regarding the quality of paths such as the end-to-end latency and packet drop rate. This information are available only at the transport layer and any solution at the IP layer has to either interact with the transport layer to obtain this information or redundantly implement some of transport layer features which are necessary to measure these values.

To the best of our knowledge, the only solution at the IP layer is proposed at (Phatak, et al., 2002). It uses IP-in-IP encapsulation to concurrently transfer data through multiple paths, transparent from the upper layers. The authors then derive the conditions which are necessary to avoid unnecessary retransmit timeouts in TCP. The proposed solution assumes that the end-to-end delays are dominated by the transmission delay of the network interface cards and hence cannot be used for the paths whose delay is dominated by propagation delay or the paths with dynamic latency and bandwidth.

Multipath routing protocols can also be placed in this category since the intermediate routers which determine a path for the outgoing packets operate at the IP layer. The proposed approaches to tackle the reordering issue caused by multipath routing mostly manipulate the congestion control mechanism of TCP (Blanton, et al., 2002)(Bohacek, et al., 2003)(Gerla, et al., 2002).

3 Approaches for TCP and UDP

In this section, we present two novel end-to-end streaming mechanisms to transfer packets of a flow through multiple paths; one for TCP connections and another for UDP flows. The proposed approach for TCP significantly reduces the number of unwanted fast-retransmit/recovery and timeout events by properly splitting of data over multiple paths at IP layer. Our approach for UDP schedules transmission of packets over multiple paths in such a way that they are received at the destination in-order, while imposing the minimum overall delay on the receiver's application.

It is worth noting that we concentrate here only on the end-to-end multipath scheme. This is different from those approaches that split a flow through multiple paths at intermediate routers, such as the one proposed in (Zhang, et al., 2003). In other words, in our scheme the sender is entirely responsible for managing multiple paths and transferring its packets through them.

We first present the assumed system model. In our network model the path latency is not assumed to be dominated by neither bandwidth nor propagation delay. Then, we propose two new approaches for multipath transferring via TCP and UDP. The proposed approaches require solely slight modification at the IP layer. We also propose analytical models to measure the effectiveness of the proposed approaches.

3.1 Network Model

In the rest of section, the model similar to the one used by (Phatak, et al., 2002) will be considered. Assume two network nodes *A* and *B*, where node *A* has some amount of data towards node *B*. Moreover, assume that there exist *n* distinct paths from *A* to *B*. We use d_i notation to indicate the average one-way delay of path *i* (i.e. propagation plus queuing and transmission delays). Without loss of generality, suppose that all paths are sorted with respect to their delays, i.e. $d_i < d_j$ $(\forall i, j, \ i < j)$.

In our model, we separate the notions of interface bandwidth and effective bandwidth. The interface bandwidth of path *i* (represented by B_i^{int}) is accounted as the

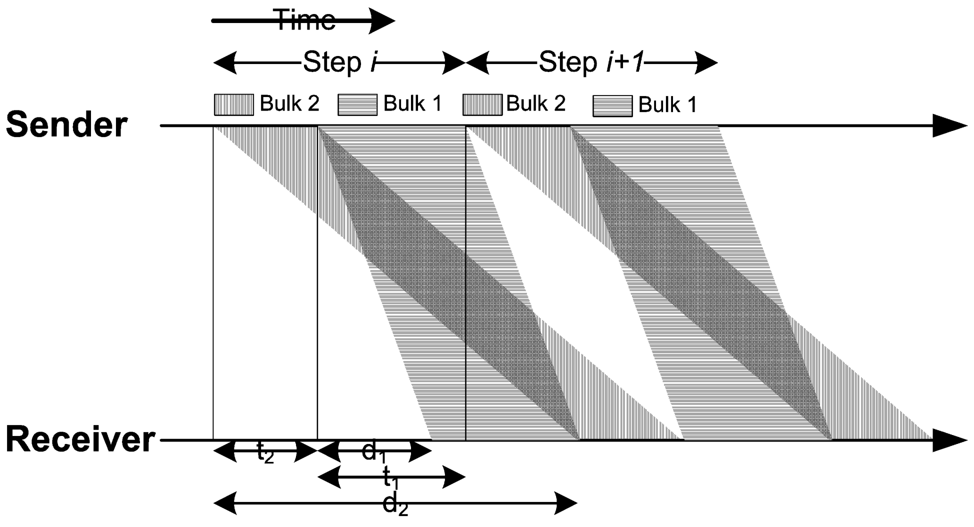


Figure 3-1: Timeline of sequential transmission steps in the presented model. The transmission time of bulk j lasts t_j seconds and the delay of path k is d_k.

bandwidth of the interface connected to the sender on path *i*. On the other hand, the effective bandwidth (referred to by B_i) represents the bandwidth that could be achieved through path *i* considering the limitation of the intermediary network between *A* and *B*. This implies that $B_i^{\text{int}} \geq B_i$. For simplicity, hereafter we use *bandwidth* term to refer to the effective bandwidth.

We also assume that the approximate values of paths' bandwidth and delay are known to the network (IP) layer. This information could be acquired by a simple extension to some link-state routing protocol such as OSPF (such as the ones proposed in (Apostolopoulos, et al., 1999) and (Liao, et al., 2001)). Another method to realize this assumption is to keep track of on-going/receiving data/ACK packets at the sender to estimate the bandwidth and delay values of paths. Note that in UDP based connections, the control packets of RTCP can play the role of ACK packets. However, this method needs some minor changes in network layer of the sender to capture the characteristics of data/ACK packets. In both TCP and UDP methods, the IP layer of the sender is responsible for splitting traffic between the available paths. This approach is of interests, because it omits the need for changing transport layer protocols like TCP and UDP.

As Figure 3-1 illustrates, the transmission process of a connection in our model is divided into some fixed-size steps. Formally, we define a *transmission step* as a part of transmission process in which all of the paths are participated exactly once (namely, every path carries a continuous set of data in each step). We use *bulk j* to indicate the amount of data which is transmitted over path *j* in each step. Generally, in each step we stream data bulks of slower paths (i.e. the paths with larger delays) sooner than those of faster ones. As we will show in the next section, operating in this manner allows us to schedule packets among multiple paths in a way that they arrive at the destination in-order. Consider Figure 3-1 again. Suppose that there exist only two available paths 1 and 2 while path 2 is the slower one. Step *i* starts with transmitting through path 2. This transmission lasts t_2 seconds. After that, the packets are carried over path 1 for t_1 seconds. With completion of transmission over path 1, the i+1th step starts with transmission through path 2 again.

It is worth noting that in this model all sending (receiving) times are measured based on the exit (entrance) of packets from (to) IP layer. This justifies the serial transmission scheme which is depicted in Figure 3-1, despite the existence of multiple network interfaces at the end hosts. The IP layer of the sender is responsible for splitting traffic between the available paths. In other words, during each step which lasts t_{step} seconds, the IP layer sends the incoming packets over path *i* for t_{i} seconds which comprise the *i*th bulk in that step. We define the effective bandwidth of path *i* (referred to by B_i) as the amount of used bandwidth by the sender through path *i*, considering the limitation of the intermediary network between *S* and *D*. In our model, we utilize paths according to their effective bandwidths, namely:

$$f_i = \frac{t_i}{t_{step}} = \frac{B_i}{B}, \qquad (1)$$

in which f_{i} is the ratio of bandwidth utilization over path *i* and *B* is total effective bandwidth over all paths.

Logically, the time needed to receive a packet from an arbitrary path is a function of packet size, interface bandwidth, and latency of that path. More specifically, the one-way delay of the i^{th} packet with size p_i sent from A to B along path k at time t is:

$$d_k^{AB}(p_i,t)=\frac{p_i}{B_k^{\text{int}}}+l_k^{AB}(p_i,t), \quad (2)$$

in which l_k^{AB} is the latency of k^{th} path between A and B (i.e. propagation plus queuing delays). Specifically, in the case of TCP connections, the RTT of the i^{th} packet can be computed by:

$$r(p_i,q_i,t)=d_k^{AB}(p_i,t)+\delta_i+d_{k'}^{BA}\left(q_i,t+d_k^{AB}(p_i,t)+\delta_i\right), \quad (3)$$

where δ_i is the delay between when i^{th} packet was received and when the corresponding ACK was sent, and q_i is the size of the ACK packet.

For the sake of simplifying our analysis, the following assumptions are made:

- All data packets have the same size p. This allows the packet dependency of delay to be removed from (2) and (3).
- We consider average latency of each path (instead of its instantaneous latency) to compute delay of packets sent over it. This allows the time dependency of delay to be removed from (2) and (3). Based on these two assumptions, we use d_k to denote the one-way delay of a packet (with size p) over path k during a given step. Nevertheless, we provide an analytical probabilistic model in section 6 to analyze our TCP-based method in networks with dynamic delay conditions.
- There is no delay between the time that a data packet is received and the time that its corresponding ACK is sent, namely $\delta_i = 0$ in (3).
- All ACKs are sent back over the same and single path to the source (e.g. the path on which the TCP SYN packet has been initially sent). We also assume that ACK packets are much smaller than data packets. Hence, the latency of

transmission for all ACKs can be assumed to be the same value Δ_{ack}. Note that $\Delta_{ack} = 0$ for UDP connections.

Based on these assumptions, for the purpose of computing the RTT of a packet sent along path *k*, we can rewrite (3) as:

$$r_k = d_k + \Delta_{ack}. \qquad (4)$$

3.2 The proposed method for TCP connections

As mentioned in Section 1, in the case of TCP, concurrent transferring of data over multiple paths can sometimes worsens the performance than using a single path (Phatak, et al., 2002). This may happen because of the following reasons:

(i) The bandwidth/delay mismatch between the selected paths causes that the sender-side TCP times out for packets sent on the slower paths and retransmits them. In addition, after each timeout, TCP drastically scales down its congestion window to zero and invokes the slow-start mechanism, thereby underutilizing the paths and further degrading the performance of multipath scheme.

(ii) Most TCP implementations include the “fast-retransmit/recovery” mechanism (Jacobson, April 1990). Even if the TCP timeout values are set high to prevent the scenario described above, fast-retransmit may still incur another serious problem. Here, we bring a simple example to elucidate this issue. Suppose the ratio of the RTTs of two employed paths is 4 and further assume that the first packet is transmitted on the slower path and the next four packets are transmitted over the faster path. The receipt of packets 2, 3, and 4 at the destination will cause the receiver-side TCP to signal a fast-retransmit of the first packet, thereby wasting the prior transmission on the slower path. Worse yet, the recovery phase following a fast-retransmit scales down the congestion window to the half of its current size, resulting in a high degradation in the throughput of the connections.

In subsequent sections, we first derive the general circumstances that could be tolerated to prevent timeouts during the lifetime of a TCP connection. Then, we discuss the conditions that should be held to avoid invocation of fast-retransmit mechanism.

3.2.1 Preventing TCP Retransmission Timeouts

In popular TCP Reno, the retransmission timeout (RTO) for the i^{th} packet is set as:

$$RTO_i = R_i + K \cdot V_i, \qquad (5)$$

where R_i is the current smoothed estimate of RTT, V_i is the current smoothed estimate of the deviation in RTT, and K is a constant factor (typically $K = 4$) which adjusts the measured retransmission timeout with respect to its variance (Phatak, et al., 2002). Moreover, R_i and V_i are calculated by the following recursive equations:

$$R_i = \psi \cdot R_{i-1} + (1-\psi) \cdot RTT_i, \quad (6)$$

$$V_i = \beta \cdot V_{i-1} + (1-\beta) \cdot \left| RTT_i - R_i \right|, \quad (7)$$

where RTT_i is the sampled RTT for the i^{th} packet. Indeed, Equations (6) and (7) act as a low-pass filter on the sampled RTT, smoothing out the variations (Mankin, et al., 1991).

According to the above discussion, the exact value of RTO will closely depend on the sequence of RTT samples. In multipath transferring schemes, this dependency is based on the sequence of paths which are chosen to send packets on. As an approximation, the average RTT and the average deviation in RTT (both weighted according to the portion of traffic which is conveyed through each path) will be used to compute RTO (Phatak, et al., 2002). The average RTT is hence calculated by:

$$\bar{r} = \sum_{i=1}^{n} r_i \cdot f_i, \qquad (8)$$

in which r_i is the smoothed estimate of RTT for packets sent along path i and f_i is the portion of traffic sent over path i. Similarly, the average deviation in RTT will be:

$$\bar{v} = \sum_{i=1}^{n} f_i \cdot \left| r_i - \bar{r} \right|. \quad (9)$$

From (5), no timeout will occur if:

$$r_j < \bar{r} + K \cdot \bar{v}, \quad \forall j, 1 \le j \le n. \quad (10)$$

According to (4) and (8), the average RTT over *n* paths is calculated by:

$$\bar{r} = \sum_{i=1}^{n} f_i \cdot (d_i + \Delta_{ack}) = \Delta_{ack} + \sum_{i=1}^{n} f_i \cdot d_i = \Delta_{ack} + \bar{d}, \qquad (11)$$

in which $\bar{d}$ is the weighted average of one-way delays, i.e.:

$$\bar{d} = \sum_{i=1}^{n} f_i \cdot d_i . \qquad (12)$$

Also, based on (4), (9), and (11), the average deviation in RTT for *n* paths will be:

$$\bar{v} = \sum_{i=1}^{n} f_i \cdot \left| d_i + \Delta_{ack} - \Delta_{ack} - \bar{d} \right| = \sum_{i=1}^{n} f_i \cdot \left| d_i - \bar{d} \right| . \ (13)$$

Considering (11) and (13), the inequality of (10) can be restated as:

$$d_j + \Delta_{ack} < \bar{d} + \Delta_{ack} + K . \sum_{i=1}^{n} f_i \cdot \left| d_i - \bar{d} \right| \quad \forall j, 1 \le j \le n. \ (14)$$

After simplification, (14) turns into the following relation:

$$\frac{d_j - \bar{d}}{\sum_{i=1}^{n} f_i \cdot \left| d_i - \bar{d} \right|} < K \qquad \forall j, 1 \le j \le n. \qquad (15)$$

Below, we try to rewrite (15) with respect to the difference between delays of paths. Since $\sum_{i=1}^{n} f_i = 1$, we clearly have:

$$d_j - \bar{d} = \left(\sum_{i=1}^{n} f_i \right) \cdot d_j - \sum_{i=1}^{n} f_i \cdot d_i = \sum_{i=1}^{n} f_i \cdot \left(d_j - d_i \right) = \sum_{i=1}^{n} f_i \cdot \Delta d_{j,i} \quad \forall j, 1 \le j \le n, \ (16)$$

where $\Delta d_{j,i}$ stands for $d_j - d_i$. Using (16), we can rephrase (15) as:

$$\frac{\sum_{i=1}^{n} f_i \cdot \Delta d_{j,i}}{\sum_{i=1}^{n} f_i \cdot \left|\sum_{y=1}^{n} f_y \cdot \Delta d_{i,y}\right|} < K \qquad \forall j, 1 \le j \le n. \qquad (17)$$

Inducing from the above equation, the condition for preventing timeout depends solely on the difference of paths' delays, i.e. $\Delta d_{j,i}$, and not on their absolute values.

3.2.1.1 Two path case

Let's consider more simple and also common case of multipath scheme in which merely two paths are exploited to carry the traffic between *A* and *B*, meaning that $n = 2$. Remind that according to our model, we have assumed $r_1 \le r_2$. Then, by setting $j = 1$ in (17), we will have:

$$\frac{f_2 \cdot \Delta d_{1,2}}{f_1 \cdot \left|f_2 \cdot \Delta d_{1,2}\right| + f_2 \cdot \left|f_1 \cdot \Delta d_{2,1}\right|} < K. \quad (18)$$

By taking into account $\Delta d_{1,2} = -\Delta d_{2,1}$ and $\Delta d_{1,2} < 0$, (18) will be shortened to:

$$-\frac{1}{2 \cdot f_1} < K, \qquad (19)$$

which is always satisfied, because both *K* and f_1 are positive numbers.

For $j = 2$, Equation (17) will be equivalent to:

$$\frac{f_1 \cdot \Delta d_{2,1}}{f_1 \cdot \left|f_2 \cdot \Delta d_{1,2}\right| + f_2 \cdot \left|f_1 \cdot \Delta d_{2,1}\right|} < K. \quad (20)$$

Similar to the simplification we made in (18), we can reduce (20) to:

$$f_1 \le 1 - \frac{1}{2 \cdot K}. \qquad (21)$$

Hence, if the employed TCP implementation is configurable, we could appropriately set the value of *K* to satisfy the above inequality. Otherwise, the sender should

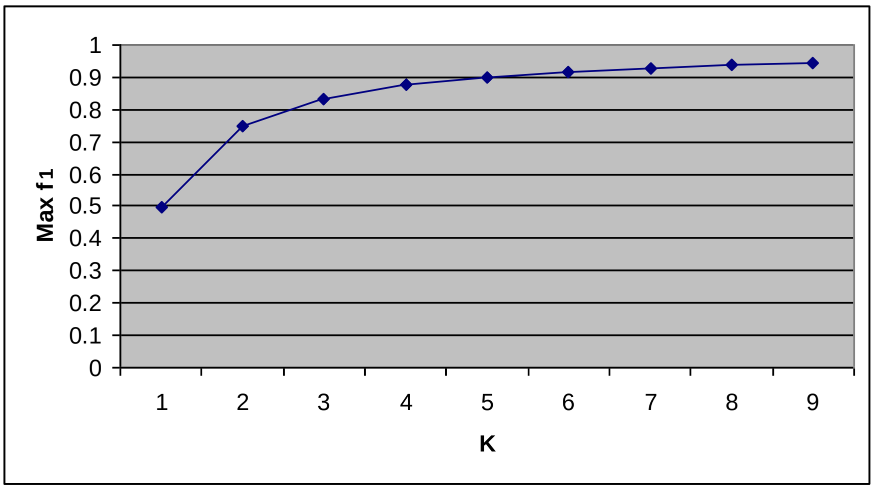


Figure 3-2: Maximum tolerable value of f_1 *under different values of K (the constant factor in TCP's timeout formula).*

carefully adapt the value of f_1 (i.e. the ratio of data sent over path 1). As Equations (19) and (21) show, the minimum value of K in two-path case (unlike the multipath case) is interestingly independent of the delay difference of paths and relates only to the bandwidth ratio of paths. Based on this condition, Figure 3-2 depicts the maximum value of f_1 that can be chosen under different values of K to incur no timeout. In common implementation of TCP Reno, the value of K is typically set to 4 (Phatak, et al., 2002). It forces us to set the value of f_1 no greater than 0.875. In other words, the sending rate over the faster path must be less than 7 times of the slower path. Although this result is the same as the one achieved by (Phatak, et al., 2002), but our model is not confined by the assumption of bandwidth domination for RTT.

3.2.2 Conditions for preventing TCP fast-retransmit

In most implementations of TCP, the fast-retransmit/recovery mechanism will be triggered if the sender receives 3 consecutive duplicate acknowledgements (ACKs). These duplicate ACKs have a very high potential to occur when using multiple paths with different delays. This is due to the fact that packets moved on the faster paths will receive sooner at the destination. Hence, they report the absence of packets which has been sent on the slower paths. The sender wrongly infers these duplicate ACKs as packet loss and falls into the fast-retransmit/recovery phase.

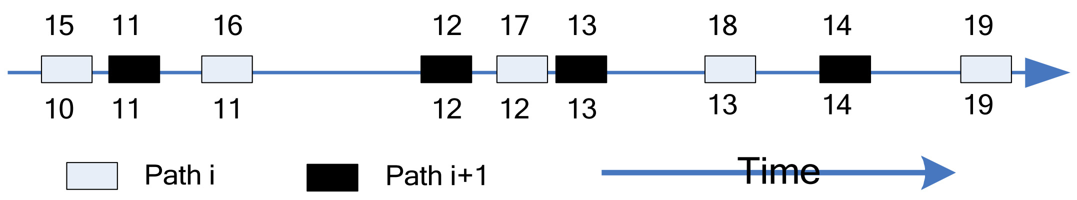


Figure 3-3: Interleaving reception of packets through two paths. With the assumption that each packet contains just one byte of data, the number above each packet represents the sequence number of that packet and the one below it stands for the last successful received sequence number.

Consider the situation in which two bulks of data come concurrently from two paths at the receiver. The basic idea is to guarantee (with a good approximation) that after reception of one packet from the faster path, the receiver will get at least one packet through the slower path. This prevents the third duplicate ACK and as a consequence, the fast-retransmit/recovery at the sender will not be triggered. This idea leads us to the concept of concurrent reception through multiple paths in which the packets from different paths would be interleaved among each other. In the case that multiple different interfaces are attached to the receiver, to realize the concurrent reception, we should slightly modify the network layer such that the packets from different interfaces are fairly scheduled and delivered to the transport layer. In other words, when multiple packets from different interfaces arrive simultaneously at the receiver, only one packet from each interface is processed at each round.

This concept is schematically illustrated in Figure 3-3. Consider a stream of data from node *A* to *B* sent over two paths *i* and *i*+1 where $d_i < d_{i+1}$. As the time proceeds, packets of both paths are received at the destination interleavingly. The number above each packet is the sequence number of the packet and the number below it stands for the ACK sequence number which is triggered by the reception of that packet. For the sake of simplicity, it is assumed that each packet contains just one byte of data. As shown in the figure, no more than two duplicate ACK is triggered per each sequence number. Despite this improvement, when delays of paths change unexpectedly, we may encounter a specific sequence of receptions which could trigger the third duplicate ACK. However, as our experimental results confirm, this situation happens rarely.

Considering start and end of reception of bulk *i* and *i*+1 in an arbitrary step, four general cases can be assumed for possible interleaving. Figure 3-4 depicts these cases. As explained earlier in this section, the dangerous duplicate ACKs are triggered only by the packets of the faster path *i*. Three possible situations could take place when a packet from bulk *i* is received: i) no packet from bulk *i*+1 has been received yet; ii) packets of bulk *i*+1 are being received concurrently with this packet; and iii) all packets of bulk *i*+1 are received before this packet. In situation ii, packets from bulk *i* and *i*+1 are received interleavingly and remind that due to concurrent reception at the receiver, it can hardly incur fast-retransmit. Hence, among these situations only the first one is dangerous (i.e. receiving some packets of bulk *i* before the reception of bulk *i*+1) which happens only in the third and forth cases of Figure 3-4. To avoid the occurrences of this situation, the length of bulk *i*+1 should be long enough to make sure that its start of reception will be before the start of reception of bulk *i*. Operating in this manner, the third and forth cases in Figure 3-4 will be converted into the first and second one, respectively.

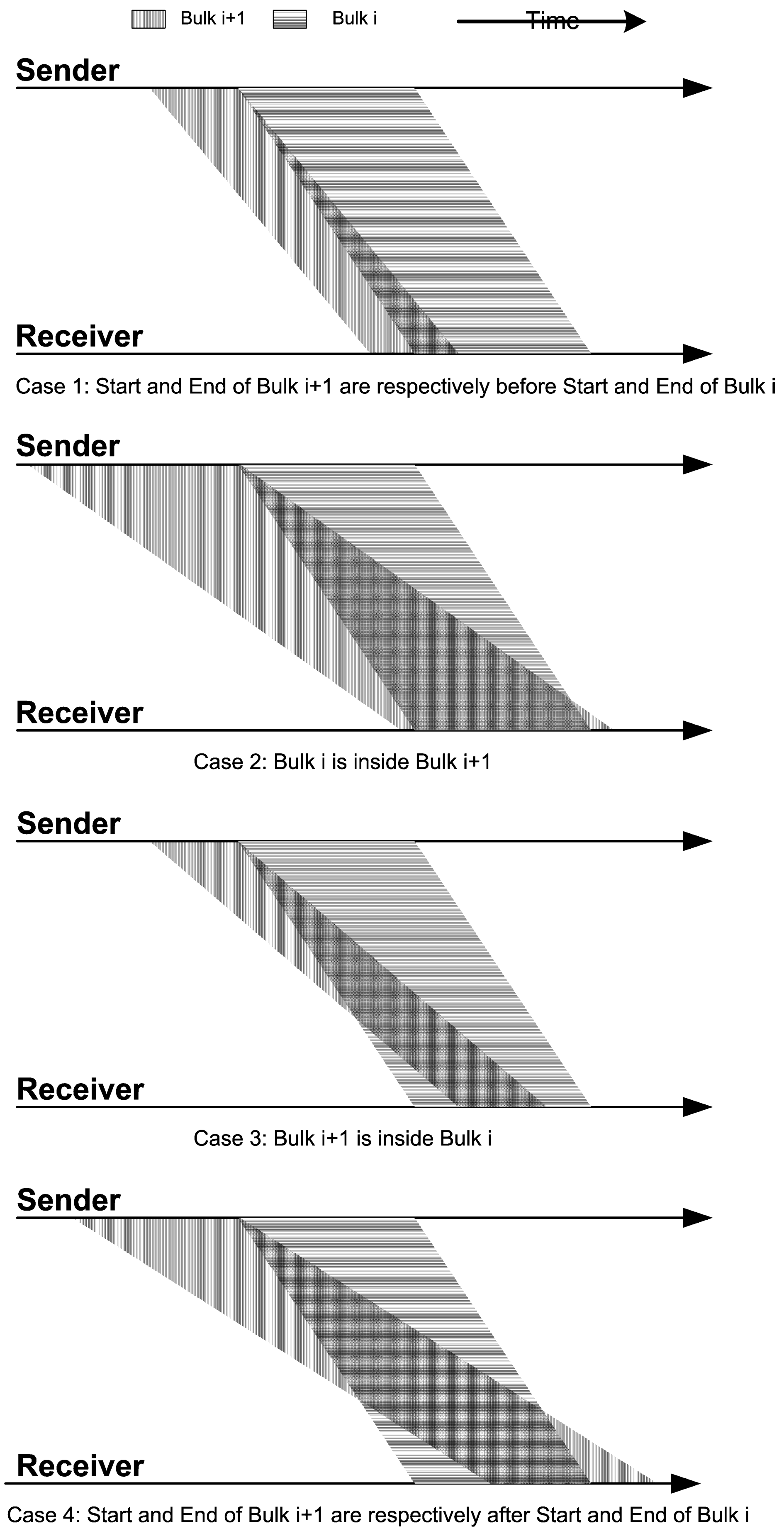
Figure 3-5 helps to understand the analytical calculations for acquiring the minimum transmission time of bulk *i*+1 (i.e. t_{i+1}). As the figure shows, we always have:

$$d_{i+1} = t_{i+1} + d_i + x, \quad 1 \le i < n. \qquad (22)$$

The value of x is the duration of the aforementioned dangerous situation. Our destiny is that the variable x has a value less than or equal to zero. This fact leads us to the following important equation which determines the minimum value of t_{i+1} to avoid fast-retransmit:

$$d_{i+1} - d_i - t_{i+1} \le 0 \Rightarrow t_{i+1} \ge \Delta d_{i+1,i}, \quad 1 \le i < n. \qquad (23)$$

Since all paths are sorted with respect to their one-way delays, we can easily conclude that (23) has a transitive property for all *i*, $0 < i < n$. In other words, if there



▥ Bulk i+1
▤ Bulk i
~~Time~~
Sender
Receiver
Case 1: Start and End of Bulk i+1 are respectively before Start and End of Bulk i
Sender
Receiver
Case 2: Bulk i is inside Bulk i+1
Sender
Receiver
Case 3: Bulk i+1 is inside Bulk i
Sender
Receiver
Case 4: Start and End of Bulk i+1 are respectively after Start and End of Bulk i

Figure 3-4: All possible overlapping cases for the reception of bulk i *and* i+1*.*

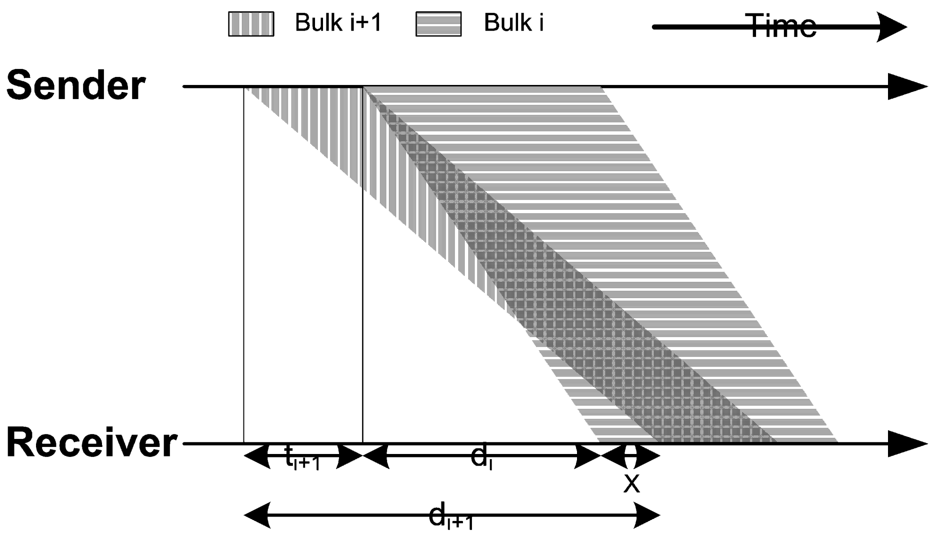


Figure 3-5: Schematic view for computing the minimum tolerable transmission time for bulk i+1.

is no interleaving between the i^{th} and i+1th paths and also between i+1th and i+2th paths, there will be no interleaving between the i^{th} and i+2th ones too. Therefore, if (23) is held for all i, $0 < i < n$, it avoids interleaving between any pair of paths i and j ($\forall i, j, 1 \le i, j \le n$).

Note that by holding the condition of (23), the 3rd duplicate ACK is avoided with a high probability. This uncertainty is due to the fact that packets sent along the same path may actually experience various delays which is different from the assumed average delay of the path. This makes the reception behavior at the destination more complex and so inspires the potential of 3rd duplicate ACK in the method. Here, we present an analytical model to obtain this probability. Consider two consecutive bulks y and y+1 (from path y and y+1, respectively) in any arbitrary step of the transmission. Since 3rd duplicate ACK happens only when two data bulks arrives at the receiver simultaneously, we take into consideration the durations in which the reception time of two bulks y and y+1 overlaps with each other. According to our previous discussion, to obtain the probability of no 3rd duplicate ACK, we only need to sum the probability of all arrival scenarios in which there is at least one packet from the slower path among every two successive packets of the faster path in

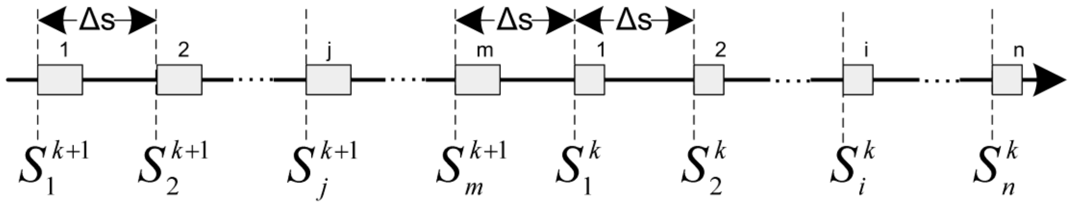


Figure 3-6: Relative order between sending times of packets belonging to two consecutive bulk k *and* k*+1.*

overlapping durations. Before that, we should explain some issues related to the receiving times of packets.

Based on (Demichelis, et al., 2002), the delay experienced by each packet over a path *y* is determined by two parameters: i) the minimum constant delay of path *y* (represented by $D_{\min}^{y}$), which can be thought as the propagation delay of that path; and ii) the additional variable delay of path *y* (denoted by D_{var}^{y}), which can be seen as the queuing and processing delay of the intermediate routers. Inspired by (Demichelis, et al., 2002) and (Corlett, et al., 2002), we assume that the additional delay of path *y* follows an exponential distribution with the average of λ^{y}. Now, let Δs be the constant interval between transmissions of two consecutive packets at the sender (Figure 3-6). The value of Δs is determined by the actual throughput of the TCP connection which remains constant despite switching between paths. Also, suppose that S_i^{y} represents the start time of transmission for the *i*th packet sent through path *y* and R_i^{y} denotes the reception time of this packet at the receiver. Then, according to above discussion, the following relations are obtained for every path *y*, $1 \le y \le n$:

$$R_i^{y+1} = S_i^{y+1} + D_{\min}^{y+1} + D_{\mathrm{var}}^{y+1}, \quad 1 \le i \le m_{y+1}, \qquad (24)$$

$$S_i^{y+1} = (i-1)\cdot \Delta s + S_1^{y+1}, \quad 1 \le i \le m_{y+1}, \qquad (25)$$

where m_{y+1} denotes the number of packets in bulk *y*+1. In addition, regarding the fact that in each step all packets of the faster path *i* are streamed exactly after those of the slower path *i*+1, we can conclude that:

$$S_i^y = (i-1)\cdot\Delta s + m_{y+1}\cdot\Delta s + S_1^{y+1} = (i+m_{y+1}-1)\cdot\Delta s + S_1^{y+1}, 1 \le i \le m_y\,. \quad (26)$$

Since all packets enter and leave the IP layer serially (not simultaneously), then the probability of equality in reception times, i.e. $\mathrm{P}(R_j^{y+1} = R_i^y)$, becomes practically zero. Thus, the probability that jth packet from path y+1 is received between ith and i+1th packets of path y with the assumption that the i+1th packet arrives after the ith one (denoted by $\mathrm{P}_{i,j}^y$) will be:

$$\mathrm{P}_{i,j}^y = \mathrm{P}(R_j^{y+1} > R_i^y) - \mathrm{P}(R_j^{y+1} > R_{i+1}^y)\,. \quad (27)$$

On the other hand, according to (24), we have:

$$\begin{aligned}\mathrm{P}(R_j^{y+1} > R_i^y) &= \mathrm{P}(S_j^{y+1} + D_{\min}^{y+1} + D_{\mathrm{var}}^{y+1} > S_i^y + D_{\min}^y + D_{\mathrm{var}}^y) \\ &= \mathrm{P}(D_{\mathrm{var}}^y - D_{\mathrm{var}}^{y+1} < D_{\min}^{y+1} - D_{\min}^y + S_j^{y+1} - S_i^y)\end{aligned}\,. \quad (28)$$

Together with (25), (26), and (28), the following relation is drawn:

$$\mathrm{P}(R_j^{y+1} > R_i^y) = \mathrm{P}(D_{\mathrm{var}}^y - D_{\mathrm{var}}^{y+1} < D_{\min}^{y+1} - D_{\min}^y + (j-i-m_{y+1})\cdot\Delta s)\,. \quad (29)$$

In order to simplify the calculations, we define threshold variable α as follows:

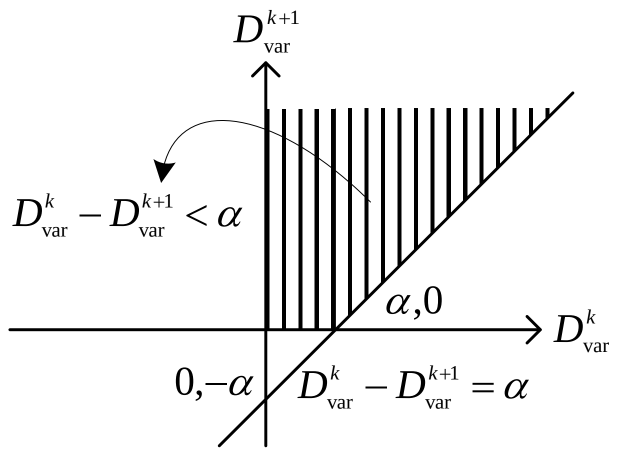
$$\alpha = D_{\min}^{y+1} - D_{\min}^y + (j-i-m_{y+1})\cdot\Delta s\,. \quad (30)$$

Then, (29) converts into:

$$\mathrm{P}(R_j^{y+1} > R_i^y) = \mathrm{P}(D_{\mathrm{var}}^y - D_{\mathrm{var}}^{y+1} < \alpha)\,. \quad (31)$$

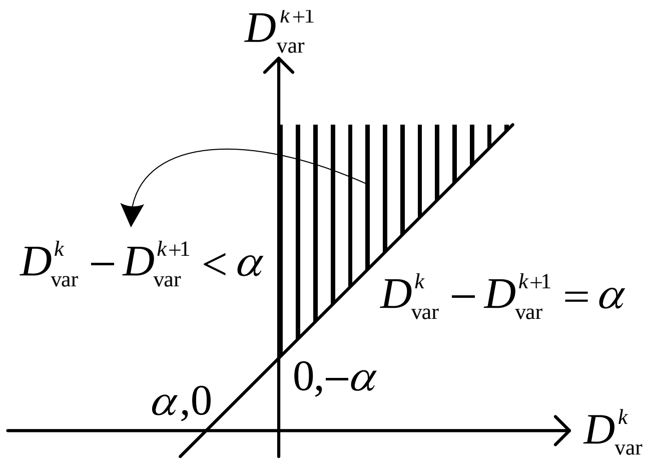
Figure 3-7 depicts the area of $D_{\mathrm{var}}^k - D_{\mathrm{var}}^{k+1} < \alpha$ in the $(D_{\mathrm{var}}^k, D_{\mathrm{var}}^{k+1})$ plane. Clearly, to calculate the probability stated in (31), we must sum the probability of $D_{\mathrm{var}}^y \times D_{\mathrm{var}}^{y+1}$ for all points in this area. As figure shows, depending on the value of α, we may encounter two different cases:

Case A) if $\alpha > 0$ then we should take into account the striped area in Case A of Figure 3-7. This leads us to the following formula:



D_{var}^{k+1}
$D_{\text{var}}^{k} - D_{\text{var}}^{k+1} < \alpha$
$\alpha,0$
D_{var}^{k}
$0,-\alpha$
$D_{\text{var}}^{k} - D_{\text{var}}^{k+1} = \alpha$

Case A) α > 0



D_{var}^{k+1}
$D_{\text{var}}^{k} - D_{\text{var}}^{k+1} < \alpha$
$D_{\text{var}}^{k} - D_{\text{var}}^{k+1} = \alpha$
$0,-\alpha$
$\alpha,0$
D_{var}^{k}

Case B) α < 0

Figure 3-7: The area in the $(D_{\text{var}}^{k}, D_{\text{var}}^{k+1})$ *plane that satisfies* $D_{\text{var}}^{y} - D_{\text{var}}^{y+1} < \alpha$. *The Case A depicts this area when α > 0 and Case B shows the mentioned area when α < 0.*

$$\begin{aligned}
&\mathrm{P}(D_{\text{var}}^{y} - D_{\text{var}}^{y+1} < \alpha) = \int_{d_{\text{var}}^{y+1}=0}^{\infty} \int_{d_{\text{var}}^{y}=0}^{d_{\text{var}}^{y+1}+\alpha} D_{\text{var}}^{y} \cdot D_{\text{var}}^{y+1} \cdot dd_{\text{var}}^{y} \cdot dd_{\text{var}}^{y+1} \\
&= \int_{d_{\text{var}}^{y+1}=0}^{\infty} \int_{d_{\text{var}}^{y}=0}^{d_{\text{var}}^{y+1}+\alpha} \frac{1}{\lambda^{y}} \cdot e^{-\frac{d_{\text{var}}^{y}}{\lambda^{y}}} \cdot \frac{1}{\lambda^{y+1}} \cdot e^{-\frac{d_{\text{var}}^{y+1}}{\lambda^{y+1}}} \cdot dd_{\text{var}}^{y} \cdot dd_{\text{var}}^{y+1} \qquad , \qquad (32) \\
&= 1 - \frac{\lambda^{y}}{\lambda^{y+1} + \lambda^{y}} \cdot e^{-\frac{\alpha}{\lambda^{y}}}
\end{aligned}$$

Case B) if α < 0 then we should consider the striped area in Case B of Figure 3-7. Thus, similar to the former case, we will have:

$$
\begin{aligned}
\mathrm{P}(D_{\mathrm{var}}^{y} - D_{\mathrm{var}}^{y+1} < \alpha) &= \int_{d_{\mathrm{var}}^{y+1}=-\alpha}^{\infty} \int_{d_{\mathrm{var}}^{y}=0}^{d_{\mathrm{var}}^{y+1}+\alpha} D_{\mathrm{var}}^{y} \cdot D_{\mathrm{var}}^{y+1} \cdot dd_{\mathrm{var}}^{y} \cdot dd_{\mathrm{var}}^{y+1} \\
&= \int_{d_{\mathrm{var}}^{y+1}=-\alpha}^{\infty} \int_{d_{\mathrm{var}}^{y}=0}^{d_{\mathrm{var}}^{y+1}+\alpha} \frac{1}{\lambda^{y}} \cdot e^{-\frac{d_{\mathrm{var}}^{y}}{\lambda^{y}}} \cdot \frac{1}{\lambda^{y+1}} \cdot e^{-\frac{d_{\mathrm{var}}^{y+1}}{\lambda^{y+1}}} \cdot dd_{\mathrm{var}}^{y} \cdot dd_{\mathrm{var}}^{y+1} \qquad , \qquad (33) \\
&= \frac{\lambda^{y+1}}{\lambda^{y+1} + \lambda^{y}} \cdot e^{\frac{\alpha}{\lambda^{y+1}}}
\end{aligned}
$$

Using (32) and (33), we are now able to compute the value of $\mathrm{P}_{i,j}^{y}$ for every *i* and *j*. Bear in mind that to compute the probability of having no 3rd duplicate ACK, we must sum the probability of all arrival scenarios of two bulks in which there is at least one packet from the slower path *y*+1 between two consecutive packets of the faster path *y* during the simultaneous reception of two bulks. To simplify our calculations, we concentrate on the most probable one, namely an arrival scenario in which the *i+1*th packet of path *y*+1 arrives at the destination between the *i*th and *i*+1th packets of path *y* for all 1 < *i* < β_y, in which:

$$\beta_y = \min(m_y, m_{y+1}) - 1. \quad (34)$$

The probability of this desired scenario (represented by P_{y}^{ok}) would be equal to:

$$\mathrm{P}_{y}^{ok} = \prod_{i=1}^{\beta_y} \mathrm{P}_{i,i+1}^{y}. \qquad (35)$$

According to (27) and (29), the probability of $\mathrm{P}_{i,i+1}^{y}$ would be:

$$
\begin{aligned}
\mathrm{P}_{i,i+1}^{y} &= \mathrm{P}(R_{i+1}^{y+1} > R_{i}^{y}) - \mathrm{P}(R_{i+1}^{y+1} > R_{i+1}^{y}) \\
&= \mathrm{P}(D_{\mathrm{var}}^{y} - D_{\mathrm{var}}^{y+1} < D_{\min}^{y+1} - D_{\min}^{y} - m_{y+1} \cdot \Delta s + \Delta s) \\
&\quad - \mathrm{P}(D_{\mathrm{var}}^{y} - D_{\mathrm{var}}^{y+1} < D_{\min}^{y+1} - D_{\min}^{y} - m_{y+1} \cdot \Delta s) \qquad , \qquad (36) \\
&= \mathrm{P}(D_{\mathrm{var}}^{y} - D_{\mathrm{var}}^{y+1} < \alpha_1) - \mathrm{P}(D_{\mathrm{var}}^{y} - D_{\mathrm{var}}^{y+1} < \alpha_2)
\end{aligned}
$$

where α_1 and α_2 are threshold variables in two probabilities of (36) which can be obtained by setting *j*=*i*+1 and *j*=*i*, respectively in (30). Since in most cases α_1 is positive and α_2 is negative, we can use (32) and (33) to compute the first and second probability of (36), respectively. To get sure about the accurate sign of α_1 and α_2 (i.e. $\alpha_1 > 0$ and $\alpha_2 < 0$) in all of the scenarios, the following constraint should be taken into consideration, when choosing the value of m_{y+1}:

$$\frac{\Delta^{d_{\min}}_{y+1,y}}{\Delta s} \le m_{y+1} < \frac{\Delta^{d_{\min}}_{y+1,y}}{\Delta s} + 1, \qquad (37)$$

where $\Delta^{d_{\min}}_{y+1,y} = D^{y+1}_{\min} - D^{y}_{\min}$. In this fashion, we reach to the following value for $\mathrm{P}^{y}_{i,i+1}$:

$$\mathrm{P}^{y}_{i,i+1} = 1 - \frac{\lambda^{y}}{\lambda^{y+1} + \lambda^{y}} \cdot e^{-\frac{\Delta^{d_{\min}}_{y+1,y} - m\cdot\Delta s + \Delta s}{\lambda^{y}}} - \frac{\lambda^{y+1}}{\lambda^{y+1} + \lambda^{y}} \cdot e^{\frac{\Delta^{d_{\min}}_{y+1,y} - m\cdot\Delta s}{\lambda^{y+1}}} . \qquad (38)$$

It can be easily understood from (38) that, $\mathrm{P}^{y}_{i,i+1}$ is not dependent to the value of *i*. Hence, based on (35) and (38), we can compute P^{ok} by:

$$\mathrm{P}^{ok}_{y} = \left[1 - \frac{1}{\lambda^{y+1} + \lambda^{y}} \cdot \left(\lambda^{y} \cdot e^{-\frac{\Delta^{d_{\min}}_{y+1,y} - m_{y}\cdot\Delta s + \Delta s}{\lambda^{y}}} + \lambda^{y+1} \cdot e^{\frac{\Delta^{d_{\min}}_{y+1,y} - m_{y+1}\cdot\Delta s}{\lambda^{y+1}}} \right) \right]^{\beta_y} . \qquad (39)$$

Note that P^{ok}_{y} provides a lower bound on the probability of having no 3rd duplicate ACK, because we haven't considered some other desired scenarios in computing the above probability. In the following, we bring an example to show how the probability of 3rd duplicate ACK can be computed according to this analytical model.

Suppose the situation where n = 2, d_1= 0.05 s, d_2= 0.225 s, $D^{k}_{\min} = 0.95 \cdot d_k$, $\lambda^{k} = 0.05 \cdot d_k$, B_1= 128 kbps, B_2= 60 kbps. Referring to our model, we straightforwardly find the following values: $\Delta^{d_{\min}}_{2,1}$= 0.166, λ^1= 0.0025, λ^2= 0.01125. To obtain the value of Δs, the packet size should be divided by throughput of the connection (i.e 188 kbps). Hence, Δs = 0.064. To satisfy the constraint of (53), m_2 should be 3 and consequently β becomes 2. These values reach us to the value of 0.83 for P^{ok}_{1}. In other words, the probability of triggering 3rd duplicate ACK in any step for two consecutive bulks is less than 0.17. In the next section, we compare the results of this probabilistic model with those achieved by simulations.

3.3 The proposed method for UDP connections

In this section, we first analyze the reordering problem of multipath transferring through a sociological perspective. Under the light of the presented model, we propose a novel end-to-end streaming mechanism to forward packets of a single flow through multiple paths. Our approach schedules the transmission of packets over multiple paths in such a way that they are received at the destination in-order while imposing the minimum overall delay on the receiver's application.

3.3.1 Sociological Perspective

In this section, we first analyze the reordering problem of multipath transferring through a sociological perspective. We show that the reordering problem in UDP is not inherently related to multipath routing; rather caused by the dominant *capitalist* perspective on the problem. Then, the problem is reconsidered through a *Marxist* perspective.

3.3.1.1 Current Sociological Perspective

Socialization is a process of learning. For the most part, socialization attempts to assure that people behave as they are expected to behave (*The Encyclopedia of Wikipedia*, s.v. "Socialization."). In fact, this is exactly the thing that is expected from packets of the routing system. This motivates us to apply the methodologies of sociology to handle the difficulties of multipath transferring. By this way of thinking, routing host represents the whole *society* and the individual packets play the role of the *individuals* of the society.

The most important notion of sociology which its correspondent in our model should be identified is the *status*. A status is a position that a person occupies within a social group (Mooney, et al., 2008). The statuses which we occupy largely define our social identity. Statuses may be either *ascribed* or *achieved*. An ascribed status is one that society assigns to an individual on the basis of factors over which the individual has no control. For example, we have no control over the sex, race, ethnic background, and socioeconomic status into which we are born. For the sake of presenting the model, the correspondents of statuses should be well defined. Moreover, we should clearly determine if they are ascribed or achieved. Considering the packets as individuals, the order of them is a good candidate to represent a status of packets because according to this order, each single packet has the priority for choosing its desired path. Plainly, this status is ascribed because the packets did not make an attempt to acquire it. A packet receives its order according to its position within the transmitting file; the fact which the packet has no control on it.

Functionalism, Conflict Theory, and Symbolic Interactionism are three common perspectives in sociology. Among them, Conflict Theory seems to be more applicable to our model. It considers societies as the competitive arenas where different groups compete for control of scarce resources (wealth, power, and prestige). In the presenting model, the scarce resources are the existing paths which have limited bandwidth. Every individual would like to use the path with minimum delay. But due to limited bandwidth of paths some packets are forced to move over the slower paths.

The dominant opinion which is ruling over some designers thoughts is that evidently the packets in front of queue for transmission should use the shortest available paths. This reminds us the way of thinking of *capitalists* who believe that people with higher degree of property could use better resources. To illustrate the similarity, let’s look at the Wilkinson definition of Capitalism: “However, space at the top of the hierarchy is scarce and a source of conflict and competition. Those who command higher status in social hierarchies have better access to material resources and mating opportunities.” (Wilkinson, 2006)

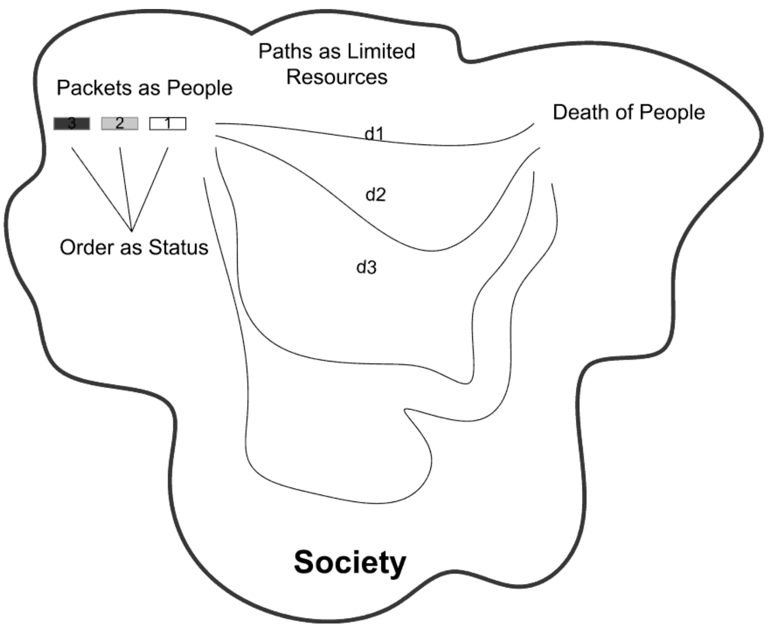


Figure 3-8: Schematic view of the presented sociological model of multipath transferring.

Figure 3-7 schematically depicts proposed sociological model of multipath transferring. Also, Table 3-1 summarizes the correspondent elements of multipath transferring and the proposed sociological model.

Multipath Routing Element	**Sociological Correspondent**
Routing Node	Society
Packets	Individuals (People)
Order of Packet	Status
Available Paths	Resources
Delays of Paths	Values of Resources
Limited Bandwidth	Limitation on Resources

Table 3-1: Correspondent elements of multipath routing and the proposed sociological model.

3.3.1.2 A Marxist Perspective

As illustrated in Figure 3-7, the existing reordering problem is a consequence of the dominant *capitalist* perspective on multipath routing, in which packets with higher orders use the shortest available paths. Consequently, later packets have to move over the slower paths. Despite the fact that the former set reaches sooner to destination, this doesn't help the overall performance of routing, since the receiver should wait for packets of the slower paths anyway. This means that the overall delay is still dominated by the slower paths delays. On the other hand, operating this like delivers packets to receiver reordered and hence causes more necessity for buffer space at the receiver.

To address this problem, let's apply the counterpart of the Capitalism in sociological perspectives (i.e. Marxism) into multipath transferring. Utilitarians and Marxists emphasized the "greatest happiness of the greatest number" and despised the desires of the individual. Mapping this notion to our model, we reach to the fact that in selecting the path, the packets with the higher orders should consider the overall benefits of transmission rather than their individual benefits. The new emerged question is how calculate the benefits of the system as whole. This question is analytically addressed in the following section.

Unfortunately, our new system inherits the drawbacks of a Marxist society; the complexity of the system is high. Perhaps the main problem of Marxist societies is how we should satisfy the individuals to ignore their own benefits in support of the society benefits. Fortunately, this does not matter here; because, packets have not the willing required to make such decisions.

3.3.2 The proposed algorithm

Consider a scenario in which we are going to split packets belonging to a single UDP connection between multiple paths. Clearly, packets which are moved over slower paths will receive later at the destination in comparison with those carried over faster paths. In general, our idea is to schedule the packets among these paths such that the destination receives them in-order. The idea is schematically depicted in Figure 3-9. The first row in the figure is sequential raw data which is ready for transmission in step *i*. The second row shows the reordered data which is actually transmitted. Finally, the last row depicts the received data at the destination. As we described

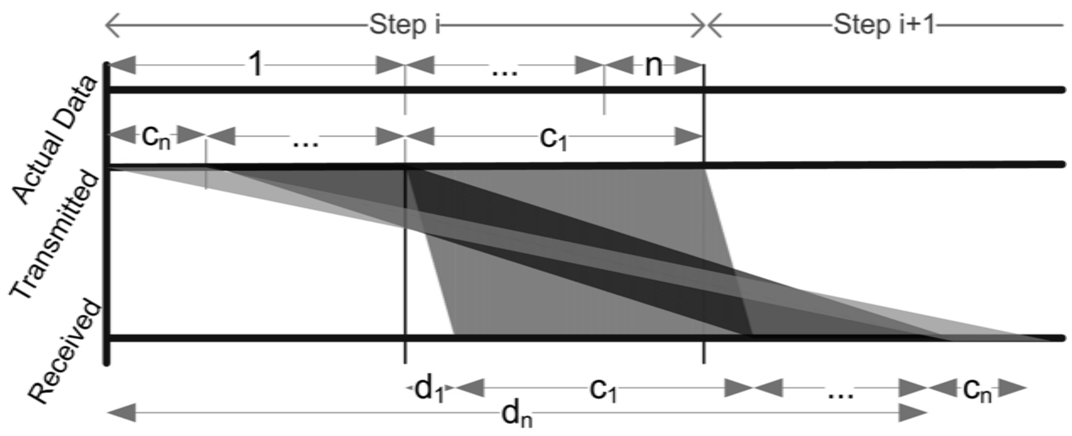


Figure 3-9: Schematic view of the proposed solution for UDP connections. The method schedules packets at the source to arrive in-order at the destination.

before, in each step the sender transmits a continuous bulk of data over every path. In the figure, each data bulk is tagged by the path number which conveys it. The intuition behind our idea is that in each step the data bulks of the slower paths are streamed sooner than those of the faster ones. Because of more delays of path j in comparison with path i ($\forall i, 0 < i < j$), the traffic could be properly scheduled so that the data bulk carried over path j is delivered at the destination after the data bulk of path i. In fact, the main issue is "how to schedule the traffic over path 1 to n in a way that the packets sent on path i ($\forall i, i > 1$) are received exactly after those carried over path *i-1*". Since the sender breaks the original order of data, the data should be completely available before the start of transmission. Consequently, this approach is not suitable for applications like VoIP and video conferencing which produces raw data instantly. However, it is very useful for some other applications like video/voice on-demand applications which usually have strong senders (e.g. powerful servers), while their receivers are from a broader range, like PDA and PC. Since the limited memory space at the hand-held devices is one of the major factors influencing design options, reducing buffer space at the receiver is much of interest.

To measure the performance of our method, we now define a practical performance metric.

Definition 1. *Beginning Pause Time* (*BPT*): the amount of time from the start of transmission (by the sender) that the receiver should pause to get sure that delivering data to the upper layer will not be interrupted.

In other word, *BPT* is the sum of buffer underrun durations for the case that the receiver starts its playing just as the sender begins its transmission. As it can be seen from Figure 3-9, the *BPT* of our solution is calculated by:

$$BPT = d_1 + \sum_{i=2}^{n} t_i \,. \quad (40)$$

To establish the scheduling illustrated in Figure 3-9, the delay of path *i* should equal the sum of the following parameters: i) the delay of path 1; ii) the elapsed time for sending traffic through path *j* for all *j*, *j* < *i* (i.e. after transmitting over path *i*); and iii) the time required to get sure that the data which will be sent after path *i* in the same step will arrive at the destination. These issues are summarized in the following equation:

$$d_i = \sum_{j=1}^{i-1} t_j + d_1 + \sum_{j=2}^{i} t_j = d_1 + 2 \cdot \sum_{j=1}^{i} t_j - t_i - t_1 \,. \qquad (41)$$

This set of independent equations along with those obtained by (1) allow us to deterministically compute the exact values of variables t_1, …, t_n. For example, consider the most practical case where n = 2, namely we use only two paths between source and destination nodes. As a consequence, (1) and (41) are rewritten as:

$$f_1 = \frac{t_1}{t_1 + t_2} \,, \quad (42)$$

$$d_2 = d_1 + t_1 + t_2 \,. \quad (43)$$

Suppose that $\Delta d = d_2 - d_1$. Based on above equations, the perfect values of t_1 and t_2 are straightforwardly computed by:

$$t_2 = f_2 \cdot \Delta d \,, \qquad (44)$$

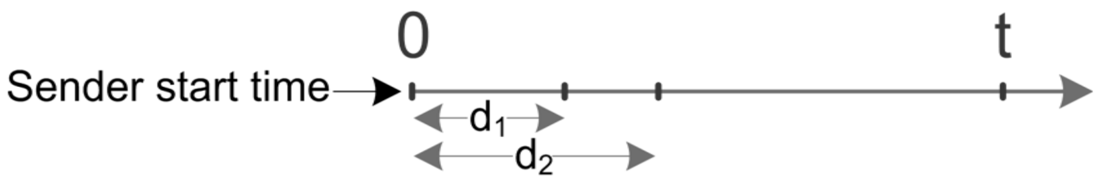


Figure 3-10: Axis of receiving times from path 1 and 2. The zero point of the axis represents the start of transmission by the sender.

$t_1 = f_1 \cdot \Delta d$. (45)

Together with (40), (44), and (45), we can conclude the following *BPT* for our solution:

$$BPT = d_1 \cdot f_1 + d_2 \cdot f_2 . \quad (46)$$

From (46), it is evident that by decreasing the amount of data which will be conveyed through the slower path, the *BPT* value will be reduced accordingly. However, we should be careful not to decrease f_2 so much that the overhead of packet header becomes intolerable.

Below, we prove that our solution delivers an optimal value for *BPT*. Assume a general streaming algorithm where the packets are received arbitrarily (i.e. in-order or out-of-order). We consider the best case in which all the received packets are in-order. Suppose an arbitrary point in the axis of time at the receiver. The zero point of the axis represents the start of transmission by the sender. Figure 3-10 illustrates this.

According to (1), the expected sending rate over path *i* would be proportional to f_i. Let *t* be a given time where $t \geq d_i$. Thus, during $[d_i,t]$, the receiver is supplied through path *i* with u_i units of time for playing, where u_i is calculated by:

$$u_i = \int_{d_i}^{t} f_i \cdot dt = f_i \cdot (t - d_i). \quad (47)$$

From definition 1, the *BPT* is equal to the sum of the idle times at the receiver in which it has no data for delivering to the upper layer, namely:

$$BPT_{opt}^{n} = t - \left[\sum_{i=1}^{n} f_i \cdot (t - d_i) \right]. \quad (48)$$

When $n = 2$ (i.e. we are using only two paths), the optimal value of *BPT* would be:

$$BPT_{opt}^{2} = t - [f_1 \cdot (t - d_1) + f_2 \cdot (t - d_2)] = f_1 \cdot d_1 + f_2 \cdot d_2 . \quad (49)$$

This value equals the value of *BPT* achieved by our solution, as shown in (46). Clearly, the *BPT* obtained by all other algorithms would be greater than or equal to this value. Consequently, the *BPT* which the receiver experiences by our algorithm is optimal.

Using Δd instead of d_2-d_1, we can rewrite (46) as:

$$BPT - d_1 = \Delta d \cdot f_2 . \quad (50)$$

Recall that d_1 is the inevitable delay of transmission between node *A* and *B*, because it is the smallest possible delay between two nodes. Hence, the above equation implies that the additional delay (i.e. *BPT*-d_1) will rise by the increase in the difference between paths' delay and also by the raise in the fraction of data which will be conveyed across the slower path. The analytical results of the *BPT* with respect to Δ*d* and f_2 are depicted in Figure 3-11. As the figure shows, the overall delay will increase multiplicatively with the raise in the fraction of data which is conveyed through the slower path.

3.3.3 Analyzing the Required Buffer Space

Now, we analyze our method with respect to the amount of buffer space required at the receiver. Based on (Demichelis, et al., 2002), the delay experienced by each packet over a path *y* is determined by two parameters: i) the minimum constant delay of path *y* (represented by $D_{\min}^{y}$), which can be thought as the propagation delay of that path; and ii) the additional variable delay of path *y* (denoted by D_{var}^{y}), which can be seen as the queuing and processing delay of the intermediate routers. Inspired by (Demichelis, et al., 2002) and (Corlett, et al., 2002), we assume that the additional

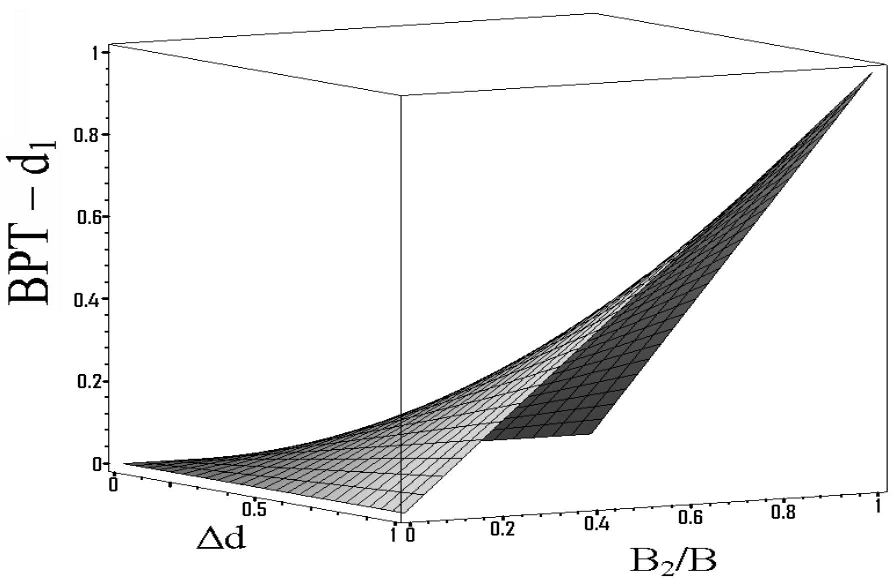


Figure 3-11: The additional delay imposed by the proposed method on the receiver's application (i.e. BPT – d1) with respect to the delay difference (i.e. Δd) and bandwidth ratio of the slower path (i.e. B2/B*).*

delay of path *y* (i.e. D_{var}^{y}) follows an exponential distribution with the average of λ^{y}. Now, let Δ*s* be the constant interval between transmissions of two consecutive packets at the sender. The value of Δ*s* is determined by the actual throughput of the connection which roughly remains constant despite switching between paths. Also, suppose that S_{i}^{y} represents the start time of transmission for the *i*th packet sent through path *y* and R_{i}^{y} denotes the reception time of this packet at the receiver. Then, according to the above discussion, the following relations are obtained for every path *y*, $1 \le y \le n$:

$$R_{i}^{y} = S_{i}^{y} + D_{\min}^{y} + D_{\text{var}}^{y}, \quad 1 \le i \le \infty. \qquad (51)$$

In addition, regarding the fact that in each step all packets of the faster path *i* are streamed exactly after those of the slower path *i*+1, we can conclude that:

$$S_{i}^{y} = S_{1}^{y''} + \left((i-1) - \sum_{j=1}^{y-1} m_{j} + \sum_{j=y+1}^{n} m_{j} \right) \cdot \Delta s, 1 \le i \le m_{y}, \ (52)$$

where m_y denotes the number of packets in bulk *y*. On the other hand, according to (51), we have:

$$\begin{aligned}\mathrm{P}(R_j^{y} > R_i^{y'}) &= \mathrm{P}(S_j^{y} + D_{\min}^{y} + D_{\mathrm{var}}^{y} > S_i^{y'} + D_{\min}^{y'} + D_{\mathrm{var}}^{y'}) \\ &= \mathrm{P}(D_{\mathrm{var}}^{y'} - D_{\mathrm{var}}^{y} < D_{\min}^{y} - D_{\min}^{y'} + S_j^{y} - S_i^{y'})\end{aligned} . \qquad (53)$$

In order to simplify the calculations, we define the threshold variable *α* as follows:

$$\alpha = D_{\min}^{y} - D_{\min}^{y'} + S_j^{y} - S_i^{y'} . \qquad (54)$$

Then, (53) converts into:

$$\mathrm{P}(R_j^{y} > R_i^{y'}) = \mathrm{P}(D_{\mathrm{var}}^{y'} - D_{\mathrm{var}}^{y} < \alpha) . \qquad (55)$$

Depending on the value of *α*, we may encounter two different cases:

Case A) if α > 0:

$$\begin{aligned}\mathrm{P}(D_{\mathrm{var}}^{y'} - D_{\mathrm{var}}^{y} < \alpha) &= \int_{d_{\mathrm{var}}^{y}=0}^{\infty} \int_{d_{\mathrm{var}}^{y'}=0}^{d_{\mathrm{var}}^{y}+\alpha} D_{\mathrm{var}}^{y'} \cdot D_{\mathrm{var}}^{y} \cdot dd_{\mathrm{var}}^{y'} \cdot dd_{\mathrm{var}}^{y} \\ &= \int_{d_{\mathrm{var}}^{y}=0}^{\infty} \int_{d_{\mathrm{var}}^{y'}=0}^{d_{\mathrm{var}}^{y}+\alpha} \frac{1}{\lambda^{y'}} \cdot e^{-\frac{d_{\mathrm{var}}^{y'}}{\lambda^{y}}} \cdot \frac{1}{\lambda^{y}} \cdot e^{-\frac{d_{\mathrm{var}}^{y}}{\lambda^{y+1}}} \cdot dd_{\mathrm{var}}^{y'} \cdot dd_{\mathrm{var}}^{y} = 1 - \frac{\lambda^{y'}}{\lambda^{y} + \lambda^{y'}} \cdot e^{-\frac{\alpha}{\lambda^{y'}}}\end{aligned} . \qquad (56)$$

Case B) if α < 0:

$$\begin{aligned}\mathrm{P}(D_{\mathrm{var}}^{y'} - D_{\mathrm{var}}^{y} < \alpha) &= \int_{d_{\mathrm{var}}^{y}=-\alpha}^{\infty} \int_{d_{\mathrm{var}}^{y'}=0}^{d_{\mathrm{var}}^{y}+\alpha} D_{\mathrm{var}}^{y'} \cdot D_{\mathrm{var}}^{y} \cdot dd_{\mathrm{var}}^{y'} \cdot dd_{\mathrm{var}}^{y} \\ &= \int_{d_{\mathrm{var}}^{y}=-\alpha}^{\infty} \int_{d_{\mathrm{var}}^{y'}=0}^{d_{\mathrm{var}}^{y}+\alpha} \frac{1}{\lambda^{y'}} \cdot e^{-\frac{d_{\mathrm{var}}^{y'}}{\lambda^{y}}} \cdot \frac{1}{\lambda^{y}} \cdot e^{-\frac{d_{\mathrm{var}}^{y}}{\lambda^{y}}} \cdot dd_{\mathrm{var}}^{y'} \cdot dd_{\mathrm{var}}^{y} = \frac{\lambda^{y}}{\lambda^{y} + \lambda^{y'}} \cdot e^{\frac{\alpha}{\lambda^{y}}}\end{aligned} . \qquad (57)$$

Consider *i*th and *j*th packet where *j=i+k* and $k > 0$. We define P_{good}^{k} as the probability of receipt of *j*th packet after the *i*th one. Accordingly, the following probabilities are defined:

$$\begin{aligned}&\mathrm{P}_{good}^{k} = \mathrm{P}(R_j^{y} > R_i^{y'}), \mathrm{P}_{bad}^{k} = 1 - \mathrm{P}_{good}^{k} , \\ &\mathrm{P}_{relative}^{k} = \mathrm{P}_{bad}^{k} / \mathrm{P}_{good}^{k} \quad 1 \le k \le \infty\end{aligned} \qquad (58)$$

For a given packet *i*, the next packets will not cause reordering, if all of them will receive after the packet *i*. The probability of this event is:

B_2/B_1 \ d_2/d_1	0.48	0.86	1.25	1.64
2	2	3	3	3
2.5	3	4	4	4
3	4	4	4	5
3.5	5	5	5	5
4	5	6	6	6

Table 3-2: Buffer length at receiver, achieved analytically for transferring a file of size 100 KB ; The rows and columns represent the ratio of bandwidth and delay of path 2 to path 1, respectively.

$$\mathrm{P}^{i}_{allgood} = \prod_{k=1}^{\infty} \mathrm{P}^{k}_{good} \quad 1 \le i \le \infty, \quad (59)$$

This yields that the maximum buffer size equals 1 for successful (in-order) delivering of the i^{th} packet to the upper layer. Generally, the probability that the maximum buffer size for successful delivery of the i^{th} packet reaches l, is the probability of reordering of l-1 individual of the packets sent after the i^{th} packet. This probability (referred to by $\mathrm{P}^{l,i}_{maxbuffer}$) can be calculates as follows:

$$\mathrm{P}^{l,i}_{maxbuffer} = \sum_{i_1=0}^{\infty} \cdots \sum_{i_l=i_{l-1}+1}^{\infty} \left(\mathrm{P}^{i}_{allgood} \cdot \prod_{k=1}^{l} \mathrm{P}^{i_k}_{relative} \right), \qquad (60)$$

Clearly, the probability that the required buffer space never exceed l throughout the whole communication, i.e. $\mathrm{P}^{l}_{maxbuffer}$, is the maximum of $\mathrm{P}^{l,i}_{maxbuffer}$ over all given values of i, namely:

$$\mathrm{P}^{l}_{maxbuffer} = \max \mathrm{P}^{l,i}_{maxbuffer} \quad 1 \le i \le \infty, \qquad (61)$$

In our network model, the lifetime of a connection is comprised from several fixed-size steps. So, we just evaluate the corresponding probabilities of packets inside a given step. Note that in the special case of one-path scenario, the probability in Equation (60) remains the same for all given values of i. For practical applications, the maximum length of buffer which will be realized in 95% of situations is of interest. Hence, the required buffer space at receivers, l, will be minimum value of k

for which $\mathrm{P}_{\max buffer}^{l,k} < 0.05$. Table 3-2 presents the obtained analytical results for our multipath method.

4 Experimental Results

This section first presents the simulation model used to evaluate the performance of the proposed methods and then provides the experimental results.

4.1 Simulation Model

For the purpose of simulation, we implemented the TCP Reno protocol in Java. To provide UDP-based connections, we also implemented the RTP/RTCP protocol. In our simulations, two threads act as a client and a server. A file will be transferred from the server to the client through a TCP (or UDP) connection. The implemented code simulates the common characteristics and limitations of real networks like dynamically changing latency of links and drop behavior when the load exceeds the link capacity. The dynamic delay follows an exponential distribution which has been described thoroughly in Section 3.2.2. Moreover, it enables the IP layer to split the outgoing traffic among multiple distinct paths.

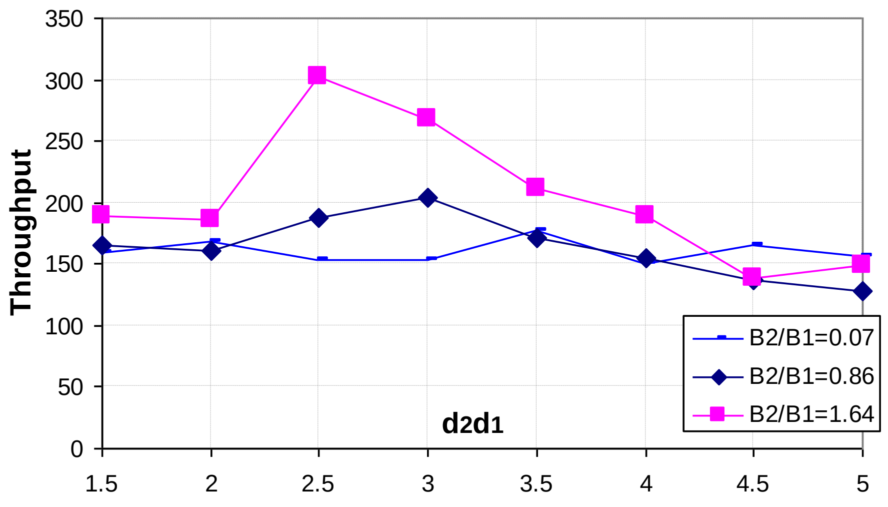
For the sake of simplicity, we just consider two available paths between the client and the server (i.e. path 1 and 2), while path 1 is the faster one. The interface bandwidth of two paths is the same (i.e. 2 Mbps) and the size of all data packets is fixed to 1.5 KB. Also, the bandwidths of path 1 and 2 are limited to B_1 and B_2, respectively. The configurable parameters of our experiments are i) the size of the transferred file; ii) the relative latency of the paths; and iii) the relative bandwidth of the paths.

Overall, three different UDP-based methods are simulated in our experiments: i) the usual one-path UDP with aggregated bandwidth of $B_1 + B_2$; ii) the simple multipath UDP which simply divides the traffic between two paths according to their bandwidth ratio; and iii) our enhanced multipath whose transmission parameters are set based on (44) and (45). Similarly, three methods are simulated for TCP: i) the usual one-path TCP with aggregated bandwidth of $B_1 + B_2$; ii) the simple multipath TCP; and iii) the multipath TCP enhanced by the techniques we proposed in this paper. Clearly, the results of the one-path scenario with aggregated bandwidth give the optimal values which could be obtained by a perfect multipath method. In the simple multipath scenario, the outgoing packets are divided at the IP layer between two available paths according to the ratio of their bandwidths (i.e. in each point of the time, path 1 has carried B_1 / (B_1 + B_2) portion of the outgoing data.) We measure the *BPT* parameter to compare performance of UDP methods. Also, the effective throughput, number of fast-retransmit events, number of timeouts and number of dropped packets are our performance metrics for evaluating TCP methods.

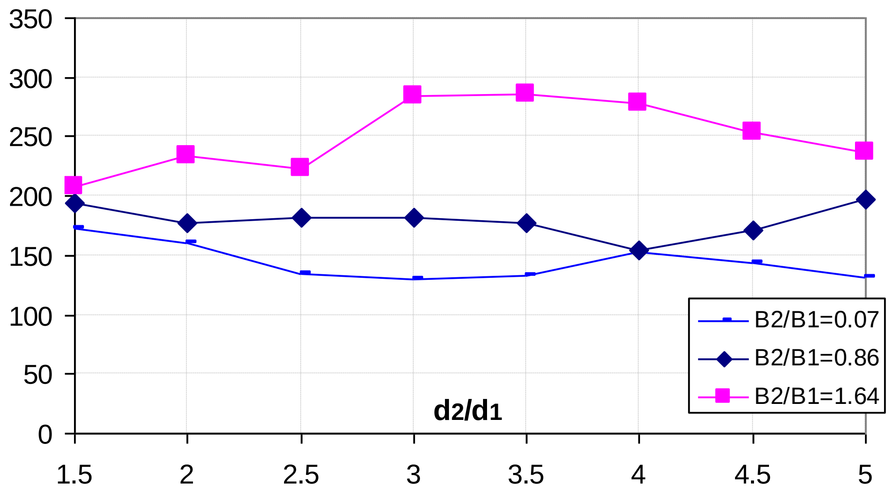
In all of the following experiments, we assume that the average latency and the bandwidth of path 1 are fixed to 0.05 s and 128 Kbps, respectively. Also, all diagrams are plotted based on the relative latency and bandwidth of path 2 with respect to those of path 1.

4.2 Simulation Results for TCP

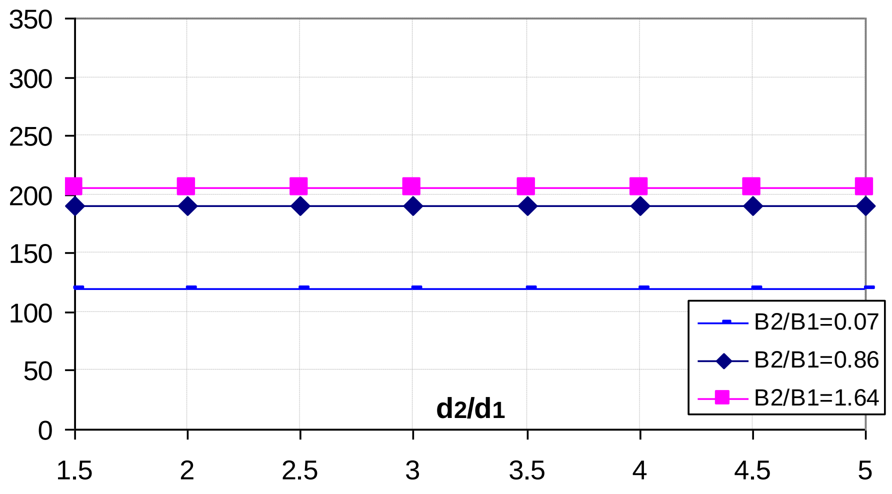
This section presents the results of various simulations we have carried out to assess our TCP-based method. As the first simulation result, Figure 4-1 shows the results of throughput obtained by different methods for transferring a file of size 100 KB. According to the figure, the throughput will improve significantly by applying our method on the multipath transferring scheme. In general, with the raise in the bandwidth of the auxiliary path 2, the throughput of all methods increases too. However, by the raise in latency of path 2, the throughput considerably decreases in the simple multipath scenario. Although this reduction exists in our approach too, but it’s slope is considerably smoothed. As a numerical example, in the case that the bandwidth of path 1 and 2 are 128 kbps and 110kbps (i.e. B_2 / B_1 = 0.86) and their delays are 0.05 ms and 0.25 ms (i.e. d_2 / d_1 = 5) respectively, the achieved throughput for transferring a file of size 100KB by our method is 197.8 kbps which is very close to the one obtained by the optimal one-path transmission (i.e. 189.8 kbps). However, this value is 128.3 kbps in simple multipath approach which is very fewer than what achieved by the former methods.



(a) Simple multipath method



(b) Our enhanced multipath method



(c) One-path method

Figure 4-1: Throughput achieved by TCP methods for transferring a file of size 100 KB.

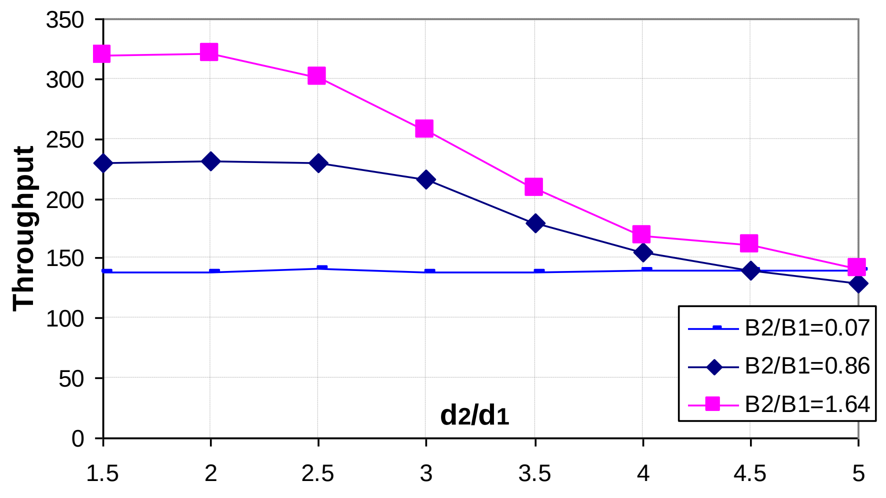
RTT and also the throughput of the path. During this time, all methods are in their transient state in which their behaviors become somehow complicated.

Surprisingly, in some situations in Figure 4-1, our method outperforms the one-path method in terms of throughput. To analyze this observation, we should note that the results depicted in Figure 4-1 are obtained by transferring a file of small size (i.e. 100 KB) while the methods are often in their transient state. It means that the TCP parameters (e.g. timeout) have not been stabilized yet. In this situation, the timeout value in our method is more influenced by the delay of path 2 at the beginning steps of transmission, because transmission starts with bulk 2. In consequence, the timeout value is greater than that of the one-path method in transient states. On the other hand, according to Figure 4-3, the number of dropped packets in the one-path method is considerably more than our method. It means that the drop probability is higher in the one-path method. Keeping these two parameters in mind, let's look at the TCP performance formula (Padhye, et al., 2000):

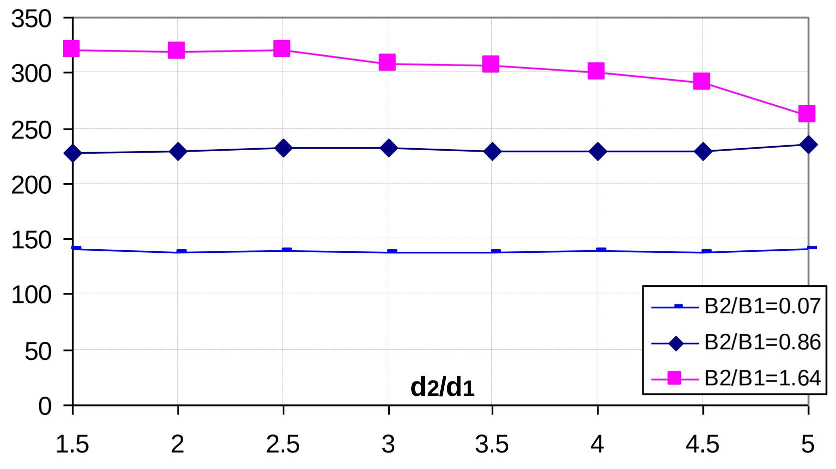
$$\hat{\lambda} = \frac{1}{\sqrt{\frac{2bp_l}{3}} + \frac{T_0}{R}\min(1,3\sqrt{\frac{3bp_l}{8}})p_l(1+32p_l^2)} packets/RTT, \quad (62)$$

where λ is the throughput, p_l is the packet loss rate of a TCP streaming flow, R is the round-trip time, T_0 is the retransmission timeout, b = 2 if delayed ACK is implemented at the receiver, and otherwise b = 1. Based on this relation, the throughput of TCP generally improves by decreasing each of the drop probability p_l and timeout duration. However, the exact throughput of TCP in different situations depends on the compromise between these two parameters. During the initial transient state, our enhanced multipath approach has a smaller drop probability while the one-path method has a less timeout duration. As indicated before, these complex behaviors are observed only in transient states. When the file size is large enough, these behaviors disappear completely.

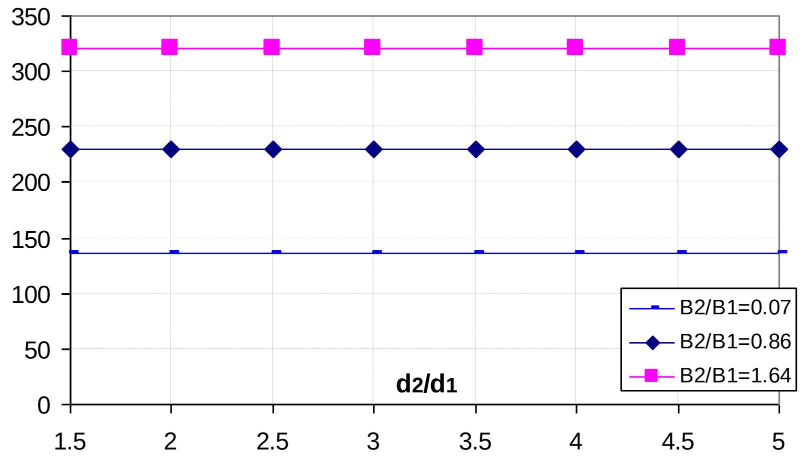
Figure 4-2 depicts the resulted throughput for transferring a file of size 1000 KB by different methods. Still, the enhanced multipath approach shows very good results, as



(a) Simple multipath method

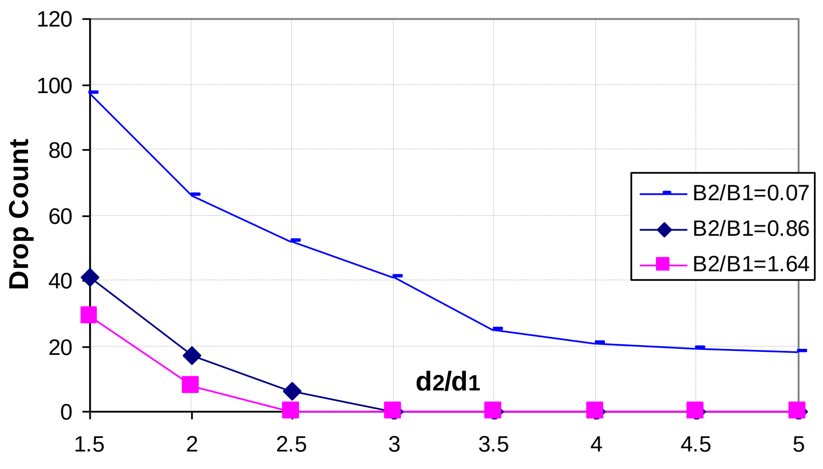


(b) Our enhanced multipath method

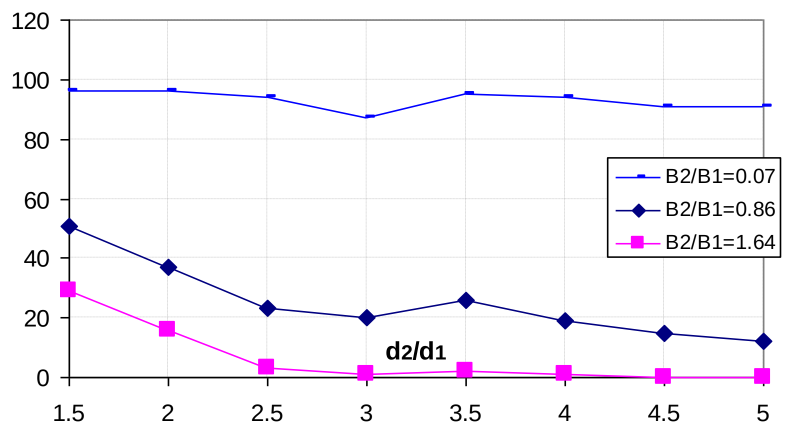


(c) One-path method

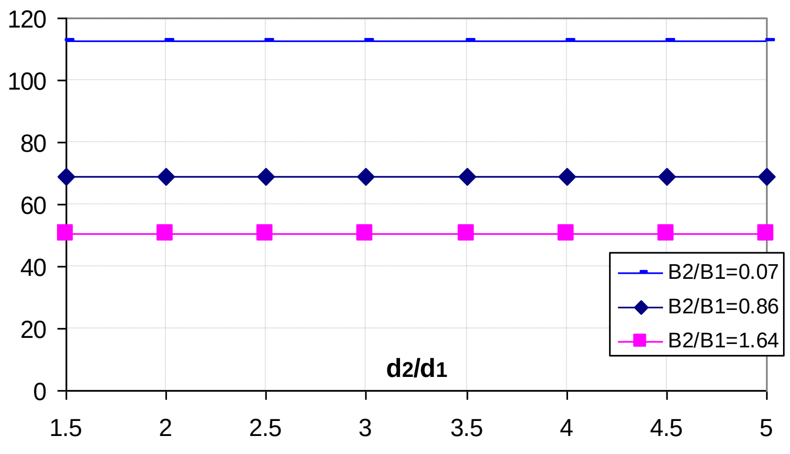
Figure 4-2: Throughput achieved by TCP methods for transferring a file of size 1000 KB.



(a) Simple multipath method



(b) Our enhanced multipath method



(c) One-path method

Figure 4-3: Number of drops in TCP methods for transferring a file of size 1000 KB.

delays are 0.05ms and 0.20ms (i.e. d_2 / d_1 = 4) respectively, the achieved throughput for transferring a file of size 1000 KB by our method is 229.8kbps which is very close to the one obtained by the optimal one-path transmission (i.e. 230.7 kbps). However, this value is 155.9kbps in simple multipath approach which is very fewer than what achieved by other two methods.

The number of dropped packets is illustrated in Figure 4-3. This parameter is different from fast-retransmit in a sense that here fewer number of drops does not necessarily imply a better performance. It may have another implicit meaning that TCP has fought less to achieve the maximum bandwidth.

Figure 4-4 depicts the results of dropped throughput in our approach with respect to the one-path transferring method. Moreover, the analytical probability of triggering 3rd duplicate ACK calculated by Equation (39) is demonstrated in Figure 4-5. Comparing these two diagrams with each other, we can easily justify the behavior of our method in terms of dropped throughput. Ignoring the local changes, both dropped throughput and probability of 3rd duplicate ACK increase by the raise in bandwidth and delay of path 2. In other words, the main reason for the increase in dropped throughput of our method (when D_2 and B_2 increase) is the occurrence of large number of 3rd duplicate ACKs in such conditions.

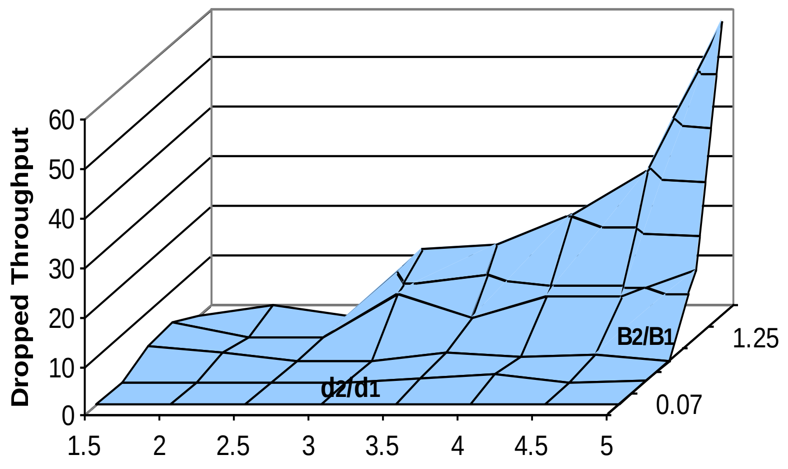


Figure 4-4: Dropped throughput in our approach with respect to the one-path method. These results are measured by transferring a file of size 1000 KB.

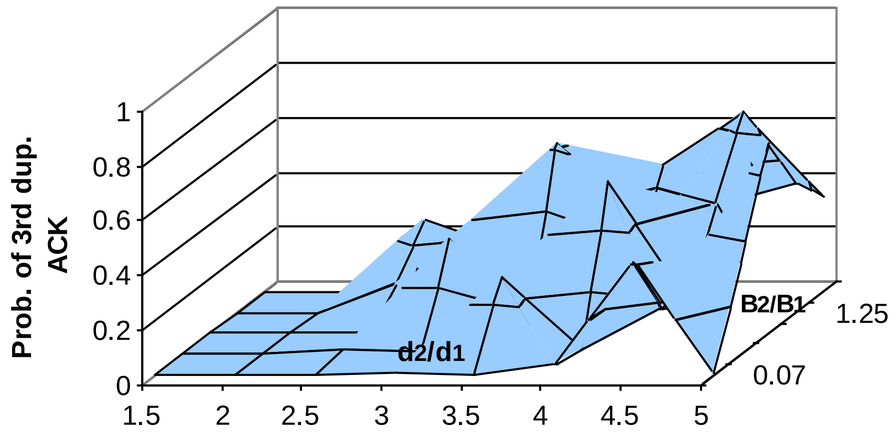


Figure 4-5: Analytical results for the probability of triggering 3rd duplicate ACK computed through Equation (39).

Table 4-1 compares the number of fast-retransmit events in all methods. The rows and columns represent the ratio of bandwidth and delay of path 2 to path 1, respectively. In this table, the file size is fixed to 1000 KB. As the table shows, the rate of fast-retransmit events decreases considerably in our approach and approximately reaches to the results of the one-path method. As an example, consider the case where the bandwidth and the delay of path 2 are equal to 110kbps (i.e. B_2 / B_1 = 0.86) and 0.25 s (i.e. d_2 / d_1 = 5), respectively. By our method, the number of fast-retransmit/recovery events which is 9 units by the one-path method increases just by 5 units. This increase in simple multipath approach reaches 30 units which unfortunately could not be tolerated.

B_2/B_1 \ d_2/d_1	0.07			0.86			1.64		
	SM	**EM**	**OP**	**SM**	**EM**	**OP**	**SM**	**EM**	**OP**
2	23	11	5	21	14	9	14	12	7
2.5	26	11	6	26	18	9	17	15	6
3	34	11	7	30	17	9	19	15	8
3.5	47	8	6	36	13	10	25	14	7
4	61	10	7	38	13	9	30	13	8
4.5	62	9	6	39	15	10	27	13	8
5	65	10	5	39	14	9	30	15	7

Table 4-1: Number of fast-retransmit/recovery events for transferring a file of size 900 KB; The numbers in each cell belong to the number of fast retransmissions obtained respectively by simple multipath (SM) approach, enhanced multipath (EM) approach, and one-path (OP) method. The rows and columns represent the ratio of bandwidth and delay of path 2 to path 1, respectively.

Finally, the number of timeout events in our method is schematically drawn in Figure 4-6. In this experiment, the file size is 1000 KB. As we expected beforehand from (37), the number of timeout events is nearly independent of the delay ratio of two paths. Moreover, the rate of timeout events increases hugely when bandwidths of the paths do not satisfy the condition of (21) (i.e. when B_2 / B_1 is less than 0.14).

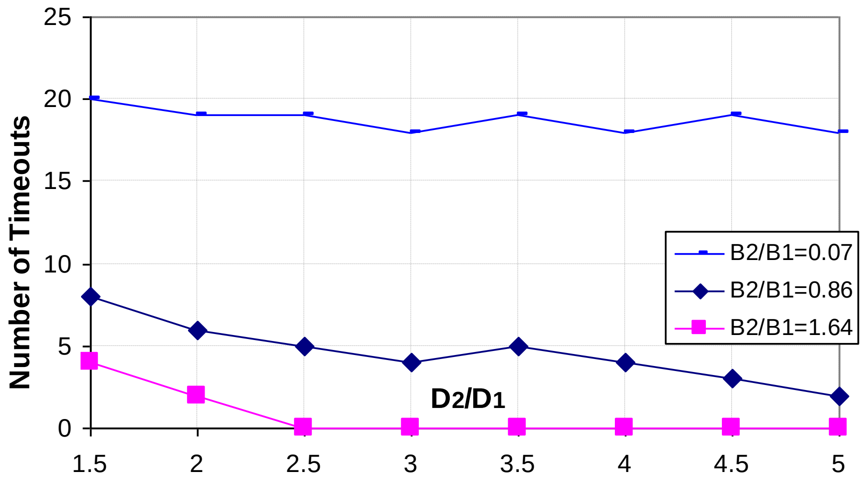


Figure 4-6: Number of timeout events in our method for transferring a file of size 1000 KB.

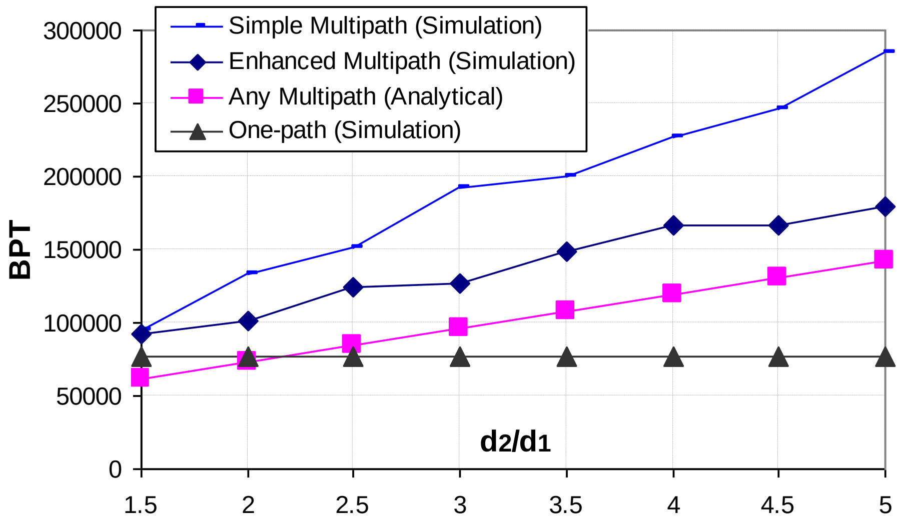
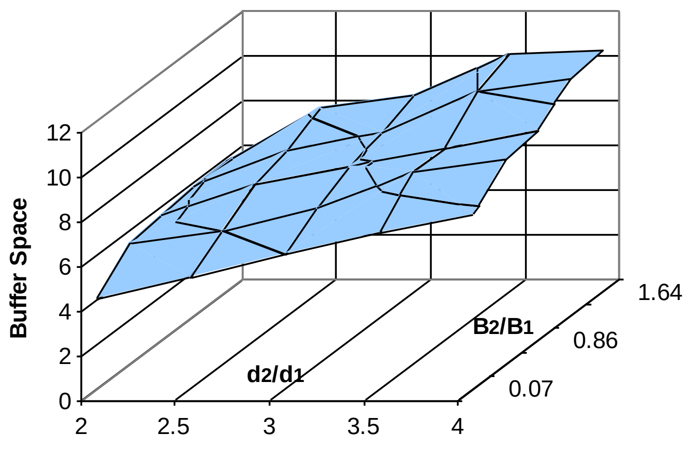


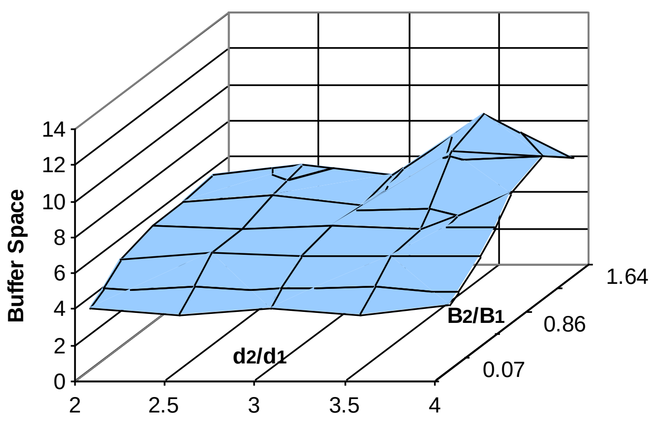
Figure 4-7: BPT achieved by UDP methods for transferring a file of size 100 KB.

4.3 Simulation Results for UDP

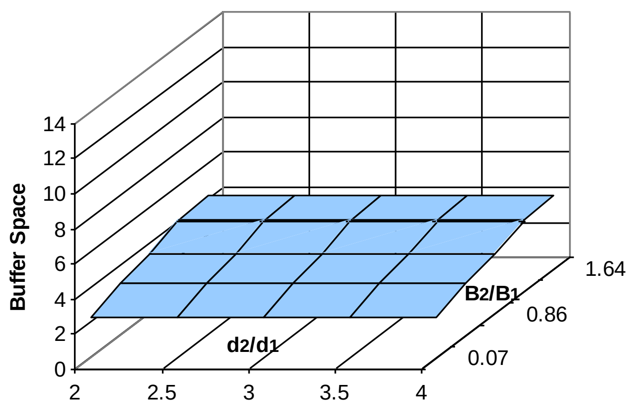
Here, we compare all three UDP methods with respect to the BPT parameter. Figure 4-7 shows the experimental results of BPT obtained by different methods for transferring a file of size 100 KB. To give the reader better insight, the figure also presents the analytical result of BPT in our approach, calculated based on (16). As we proved in Section 3.3, this result shows the minimum value of BPT that can be obtained by any multipath method. According to the figure, the one-path method completely outperforms two other multipath approaches, especially when d_2 / d_1 is



(a) Simple Multipath method



b) Proposed Multipath method



c) One-path method

Figure 4-8: The measured buffer level of three methods.

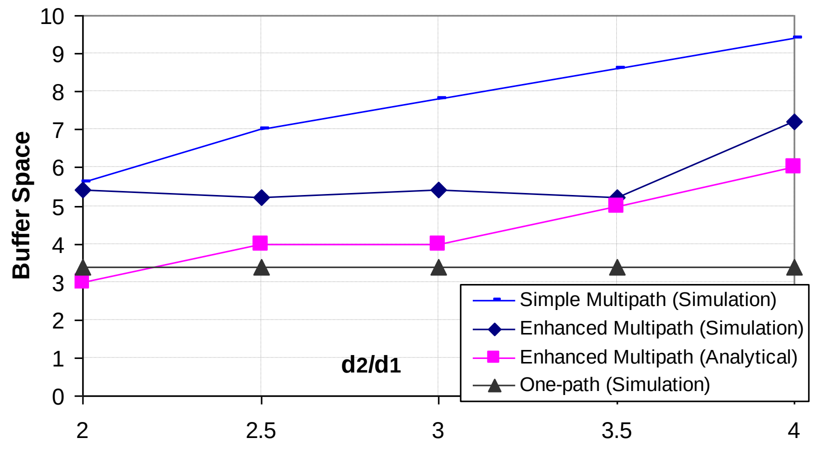


Figure 4-9: The measured buffer level of the three methods for B_2 / B_1 *equals to 0.86 through different delay ratios.*

high. However, our enhanced multipath method significantly improves the BPT, compared with simple multipath approach. As much as the delay ratio of two paths becomes higher, this improvement becomes more apparent. Also, the simulation results of our approach are on average 20% above the analytical ones, which is mainly due to the existence of packet delay variance and drop behavior in our experiments.

With the assumption that the receiver's application starts its playing with the optimal measured *BPT*, the consumed buffer space at the receiver is depicted in Figure 4-8. As illustrated in the figure, the consumed buffer space generally increases with the raise in the effective bandwidth ratio. In the case of simple multipath scenario, the buffer level grows drastically with the increase in the delay of the slower path. Although we can observe this growth in our approach too, but its variation has been well controlled now. One can easily comprehend the similarity between the analytical results, presented in the previous section, and these experimental ones. For instance, both results increase as the ratio of B_2 / B_1 becomes higher.

To better demonstrate the effect of our proposed method with respect to buffer space, a 2-D snapshot of Figure 4-8 is rendered in Figure 4-9, comparing the results of all three methods. As the figure shows, the consumed buffer level of our method is close to the inevitable level implied by the one-path method. Also, the level of buffer space remains steady throughout the increase in the delay ratio of path 2 to path 1. This is in spite of the simple multipath method that the buffer level raises drastically with the increase in the delay ratio.

5 Conclusion and future works

The main problem of multipath transferring schemes is the difference between the delays of selected paths which causes reordering between packets of the same flow. Multipath methods in both wired and wireless networks prefer more diverse paths and this unfortunately intensifies the problem of reordering. In this paper, two novel streaming approaches have been introduced for handling the reordering problem in end-to-end multipath schemes.

5.1 Reordering problem in UDP

In the case of UDP connections, we proposed a streaming method for scheduling packets at the sender among multiple paths in a way that they arrive at the receiver in-order. It has been proven that our proposal imposes the minimum possible delay on the receiver's application to start after the sender begins its transmission. In addition, our method reduces the need for large buffer spaces at the receiver. Simulation results also confirm the efficiency of applying the proposed method in multipath transmission.

5.2 Reordering problem in TCP

Unfortunately, in the case of TCP, the reordering brings more serious problems, i.e. producing more timeouts and unnecessary fast-retransmit/recovery events which degrade the throughput of TCP connections considerably. We first presented the analytical conditions that should be preserved to avoid timeouts. Interestingly, we proved that these conditions depend on the differences of one-way delays of paths not on their absolute values. Then, we introduced a novel approach to avoid unnecessary fast-retransmits/recovery events. The key observation is that the interleaved reception of the packets at the destination does not trigger the fast-retransmit/recovery timer, even though the packets are received reordered. Based on this observation, it has been suggested to continue transmission over the slower path for duration longer than the delay differences of selected paths. The performance of our method has been compared with both simple multipath approach and the usual one-path method (with the aggregated bandwidth) through simulation. The performance has been studied in terms of throughput, number of fast-retransmit events, and number of dropped packets. Simulation results show that the performance of our approach is comparable with the optimal one-path transmission in almost all scenarios.

5.3 Future works

For applications which deliver the prerecorded multimedia content via TCP, minimizing the delay sensed by the receiver's application can be a potential subject for future researches. Also, the effect of the proposed probability model for 3rd duplicate ACK on the receiver buffer size is a good candidate for future work.

6 Bibliography

Abd El Al A., Saadawi T. and Lee M. Improving Throughput and Reliability in Mobile Wireless Networks via Transport Layer Bandwidth Aggregation [Journal]. - [s.l.] : Computer Networks, Special issue on Military Communications Systems and Technologies, 2004.

Abd El Al A., Saadawi T. and Lee M. LS-SCTP: A Bandwidth Aggregation Technique for Stream Control Transmission Protocol [Journal]. - [s.l.] : Computer Communications, Special issue on Protocol Engineering for Wired and Wireless Networks, 2004.

Andersen D. [et al.] Resilient Overlay Networks [Conference] // ACM Symposium on Operating Systems Principles (SOSP 2001). - Banff, Canada : [s.n.], 2001.

Apostolopoulos G. and al. et QoS routing mechanisms and OSPF extensions. - [s.l.] : IETF RFC 2676, 1999.

Argyriou A. and Madisetti V. Bandwidth Aggregation with SCTP [Conference] // IEEE Globecom. - San Fransisco, CA : [s.n.], 2003.

Blanton E. and Allman M. On Making TCP More Robust to Packet Reordering [Journal]. - [s.l.] : ACM Computer Communication Review, 2002.

Bohacek S. [et al.] TCP-PR: TCP for Persistent Packet Reordering [Conference] // IEEE ICDCS. - Rhode Island : [s.n.], 2003.

Corlett A., Pullin D. and Sargood S. Statistics of one-way internet packet delay. - [s.l.] : IETF Internet-Draft, draft-corlett-statistics-of-packet-delays-00.txt, 2002.

Day M. [et al.] A Model for Content Internetworking (CDI) [Journal]. - [s.l.] : RFC3466, IETF, 2003.

Demichelis C. and Chimento P. IP packet delay variation metric for IP performance metrics (IPPM). - [s.l.] : IETF RFC 3393, 2002.

Ganjali Y. and Keshavarzian A. Load balancing in ad hoc networks: single-path routing vs. multi-path routing [Conference] // 23nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '04). - Hong Kong : [s.n.], 2004. - Vol. 2. - pp. 1120-1125.

Gerla M., Lee S. S. and Pau G. TCP Westwood Simulation Studies in Multiple-Path Cases [Conference] // SPECTS. - San Diego, California : [s.n.], 2002.

Haas Z. J. and Pearlman M. R. The zone routing protocol (ZRP) for ad hoc networks. - [s.l.] : IETF Internet Draft (draft-ietf-manet-zone-zrp02.txt), 1999.

Hacker T. and Athey B. The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network [Conference] // IEEE IPDPS. - Ft. Lauderdale, FL : [s.n.], 2002.

Jacobson V. Modified TCP congestion avoidance algorithm [Report]. - [s.l.] : end2end interest group mailing list, April 1990.

Johnson D. B. [et al.] The dynamic source routing protocol for mobile Ad Hoc networks. - [s.l.] : IETF Internet Draft (draft-ietf-manet-dsr-03.txt), 1999.

Krishnan R. and Silvester J. A. Choice of allocation granularity in multipath source routing schemes [Conference] // 12nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '93). - San Francisco, CA : [s.n.], 1993. - Vol. 1. - pp. 322-329.

Lee S. J. and Gerla M. Split multipath routing with maximally disjoint paths in ad hoc networks [Conference] // IEEE International Conference on Communications (ICC). - Helsinki, Finland : [s.n.], 2001. - pp. 3201-3205.

Lee S. J. and Gerla M. Split multipath routing with maximally disjoint paths in ad hoc networks [Conference] // ICC. - 2001. - pp. 3201-3205.

Liao W. H. [et al.] A multi-path QoS routing protocol in a wireless mobile ad hoc network [Conference] // IEEE International Conference On Networking. - France : [s.n.], 2001. - pp. 158-167.

Ma C. and Leung K. C. Improving TCP performance robustness in multipath networks [Conference] // IEEE LCN. - 2004.

Mankin A. and Ramakrishnan K. Gateway congestion control survey. - [s.l.] : IETF RFC 1254, 1991.

Mao S. [et al.] MRTP: a multi-flow realtime transport protocol for ad hoc networks [Conference] // IEEE Vehicular Technology. - 2003.

Marina M. K. and Das S. R. On-demand multipath distance vector routing in ad hoc networks [Conference] // International Conference for Network Protocols (ICNP). - 2001.

Marina M. K. and Das S. R. On-demand Multipath Distance Vector Routing in Ad Hoc Networks [Conference] // Proceedings of the International Conference for Network Protocols. - 2001.

Maxemchuc N. F. Diversity routing [Conference] // IEEE ICC. - San Francisco, CA : [s.n.], 1975. - Vol. 1. - pp. 10-41.

Mooney L. A., Knox D. and Schacht C. Understanding social problems [Book]. - [s.l.] : Wadsworth Pub Co, 2008. - 2 : pp. 5-9.

Padhye J. [et al.] Modeling TCP Reno performance: a simple model and its empirical validation [Journal]. - [s.l.] : IEEE/ACM Transaction on Networking, 2000. - 2 : Vol. 8. - pp. 133-145.

Perkins C. E., Royer E. M. and Das S. R. Ad hoc on-Demand Distance Vector (AODV) Routing. - [s.l.] : IETF Internet Draft (draft-ietf-manetaodv-06.txt), 2000.

Phatak D. S. and Goff T. A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments [Conference] //

21nd Annual Joint Conference of the IEEE Computer and Communications Societies, (INFOCOM'02). - New York, NY : [s.n.], 2002. - Vol. 2. - pp. 773-781.

Saltzer J., Reed D. and Clark D. End-to-end arguments in system design [Journal]. - [s.l.] : ACM Transactions on Computer Systems (TOCS), 1984. - 4 : Vol. 2.

Sivakumar H., Bailey S. and Grossman R. PSockets: The Case For Application-Level Network Striping For Data Inttensive Applications Using High Speed Wide Area Networks [Conference] // EEE Supercomputing (SC). - Dallas, TX : [s.n.], 2000.

Stewart R. and al et Stream Control Transmission Protocol [Journal]. - [s.l.] : RFC2960, IETF, 2000.

Stewart R. R. and Xie Q. Stream Control Transmission Protocol: A Reference Guide [Conference]. - 2001.

The Encyclopedia of Wikipedia, s.v. “Socialization.”.

Wei W. and Zakhor A. Robust multipath source routing protocol (RMPSR) for video communication over wireless ad hoc networks [Conference] // IEEE Int’l Conference on Multimedia and Expo (ICME 2004). - Taipei : [s.n.], 2004.

Wei W. and Zakhor A. Robust Multipath Source Routing Protocol (RMPSR) for Video Communication over Wireless Ad Hoc Networks [Conference] // ICME. - 2004.

Wilkinson . - 2006.

Wu T. and Lau R. A class of self-healing ring architectures for SONET network applications [Conference] // IEEE Trans. Commun.. - 1992. - Vol. 40. - pp. 1746–1756.

Yabandeh Maysam [et al.] A Sociological Perspective on the Reordering Problem in Multipath Routing [Journal] // International Journal of Cybernetics and Systems. - March 2007. - 3 : Vol. 38. - pp. 275-288.

Yabandeh Maysam [et al.] Reducing Buffer Space in Multipath Schemes [Conference] // 5th IEEE Consumer Communications and Networking Conference (CCNC '08). - 2008.

Yabandeh Maysam, Mohammadi Hossein and Yazdani Nasser Multipath Routing in Mobile Ad hoc Networks: Design Issues [Conference] // 12th International CSI Computer Conference. - Tehran : [s.n.], 2007.

Yabandeh Maysam, Zarifzadeh Sajjad and Yazdani Nasser Improving performance of transport protocols in multipath transferring schemes [Journal] // Computer Communications. - [s.l.] : Elsevier, 2007. - 17 : Vol. 30. - pp. 3270-3284.

Ye Z., Krishnamurthy S. V. and Tripathi S. K. A framework for reliable routing in mobile ad hoc networks [Conference] // 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03). - San Francisco, CA, USA : [s.n.], 2003.

Ye Z., Krishnamurthy V. and Tripathi S. K. A Framework for Reliable Routing in Mobile Ad Hoc Networks [Conference] // IEEE INFOCOM. - 2003.

Zhang M. [et al.] A transport layer approach for improving end-to-end performance and robustness using redundant paths [Conference] // USENIX Annual Technical Conference. - Boston, MA : [s.n.], 2004. - pp. 99–112.

Zhang M. [et al.] RR-TCP: A reordering-robust TCP with DSACK [Conference] // IEEE ICNP. - 2003.