Chapter 6

The Measurements of Performance

In the following sections we will demonstrate the performance of the ONA system that has been implemented. In Section 6.1 we present the performance of the heuristic-search-based planner tested separately from the whole system. We tested it versus an exhaustive search algorithm planning. In Section 6.2 we tested the Panda overhead and the performance of Panda is connection with real-time multimedia applications. The results of the tests show a significant advantage had by the connections that used Panda. The results with real applications also demonstrate the advantages of central planning versus incremental planning.

6.1 Performance of Heuristic Search-Based Planning

We tested a Java implementation of this planner on Dell Inspiron laptops with 333 MHz processors. Connections were generated in a random fashion. The links between the nodes were randomly assigned bandwidths of 10 Mbps, 2 Mbps, or 100 Kbps. Moving data over a 10-Mbps link required no adaptation. Moving it over a 2-Mbps link required Lempel-Ziv compression. Moving it over a 100-Kbps link required both lossy filtering and Lempel-Ziv compression. Each link was designated secure or insecure, which required no adaptation or encryption and

decryption, respectively. Therefore, each link could require at most five adaptations: filtering, compression, decompression, encryption, and decryption. We also generated a resource availability for each node, in terms of the number of adapters the node is able to run.

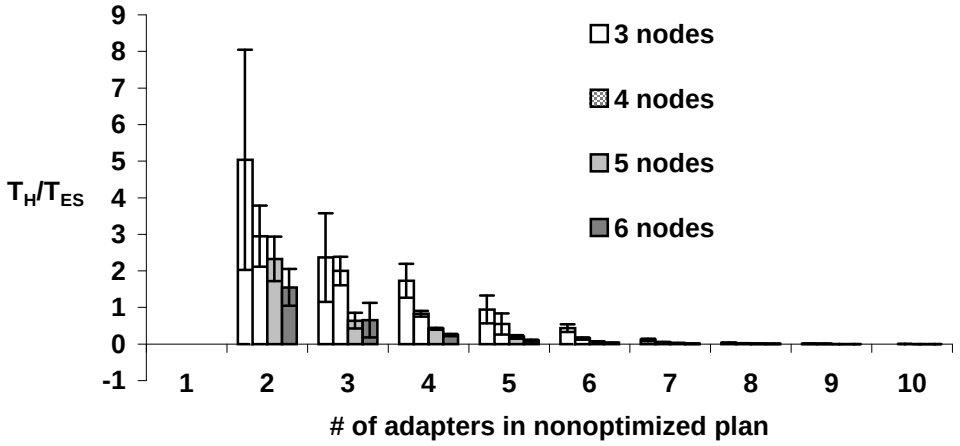


Figure 6.1: Heuristic to exhaustive search time cost ratio

We generated 1000 different scenarios of this kind, which we classify by the number of nodes in the connection and the original number of adapters chosen to handle the connection before any optimization. Binary adaptations are counted as two separate adaptations: for example, we consider Ziv-Lempel compression and decompression as two adapters. Because of the random selection of problems, unequal numbers of cases occurred for different combinations of the number of nodes and the number of adaptations; some cases occurred a lesser number of times than the others. Different cases required different solution times but they had the same number of adapters in their nonoptimized plans. These effects caused wide variations in the statistical distribution of different classifications. Exhaustive search was used to evaluate the speed and the efficiency of the heuristic search. Exhaustive search verifies all combinations of

the selected adapters on all nodes. However, the number of combinations is reduced so that each adapter always covers the correspondent problematic link that it is assigned to fix: the DO part always remains before the link and the UNDO part (if any) always remains after the link. This constraint seriously reduces the search space of potential solutions, eliminating those that cannot be effective, and giving us a more realistic comparison.

Except in the 4% of all cases where no feasible plan existed, heuristic search found some feasible plan. In one case with four nodes, three cases with five nodes, and eight cases with six nodes, (12 cases out of 1000), the heuristic search did not find the same optimal plan as exhaustive search. All these cases occurred when most connection nodes did not have enough resources to run necessary adaptations.

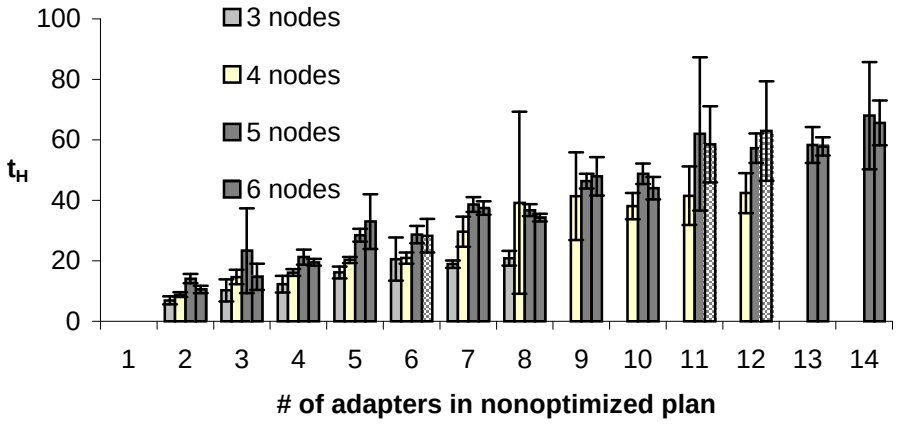


Figure 6.2: Heuristic search performance (in milliseconds)

Figure 6.1 shows the ratio between the duration of exhaustive search and heuristic search. Exhaustive search has the advantage over heuristic search for small numbers of adaptations and nodes. For any number of nodes using more than six adapters and for more than four nodes using

more than two adapters, heuristic search outperforms an exhaustive search. For cases with more than seven adapters, the exhaustive search cannot compete with the heuristic search. Closer study of the cases showed that the location of a problematic link seriously affects the number of possible adapter combinations for exhaustive search, making the confidence interval wide.

Figure 6.2 shows the performance of heuristic search for different numbers of adaptations and nodes. For cases from Figure 6.1 where exhaustive search outperformed heuristic search, the heuristic search time is relatively short, around 10 milliseconds. The worst average heuristic search time on the chart is around 60 milliseconds, which would usually be an acceptable delay, especially given the probable poor performance of the connection if no adaptation was done.

Figure 6.3 replots some of the data from Figure 6.2 to clarify how the performance of heuristic search depends on the number of nodes for a given number of adaptations. In these tests, the number of adapters shrinks with almost every step of merging; for example, only one filter, encryptor, or compressor of any two survives a merge. That is why the shape of the graphs looks close to linear, ignoring the actual complexity of the algorithm. The latency of planning did not reach 200 milliseconds for practical cases.

6.2 Costs of Incremental Versus Central Planning

As mentioned in the previous section, an incremental plan can be used for the connection if time limitations do not allow the plan's optimization. Figure 6.4 shows the planning time of incremental planning versus the corresponding central planning.

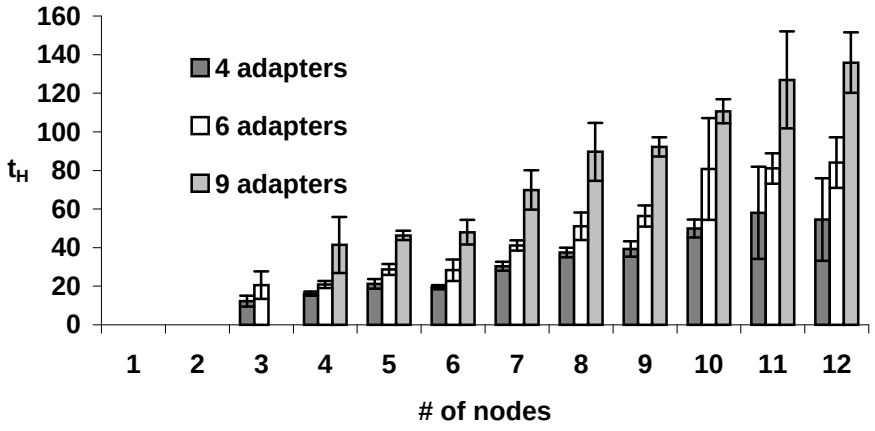


Figure 6.3: Heuristic search performance (in milliseconds)

Incremental planning takes 3 to 5 milliseconds, while central planning takes tens or hundreds of milliseconds. The difference in the efficiency of incremental and central plans is shown in Figure 6.5 in plan cost units calculated with the optimization function (the fewer units the better). As expected, optimized plans are significantly better.

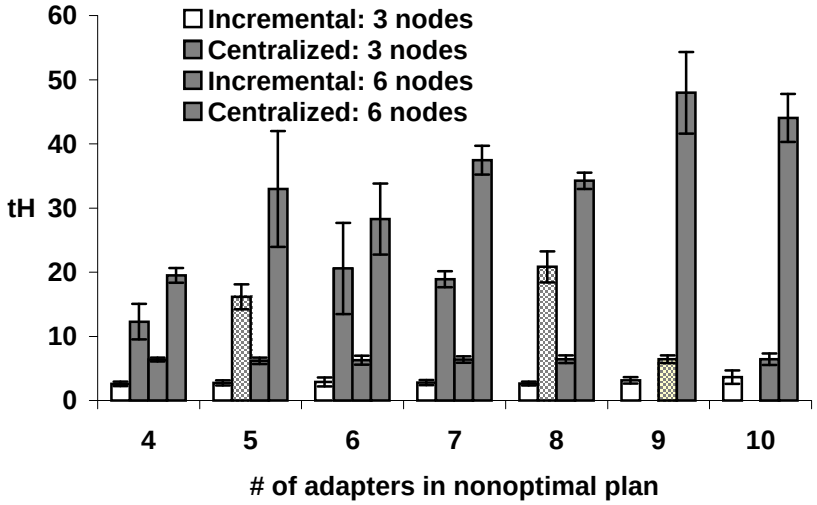


Figure 6.4: Incremental versus central planning time cost (in milliseconds)

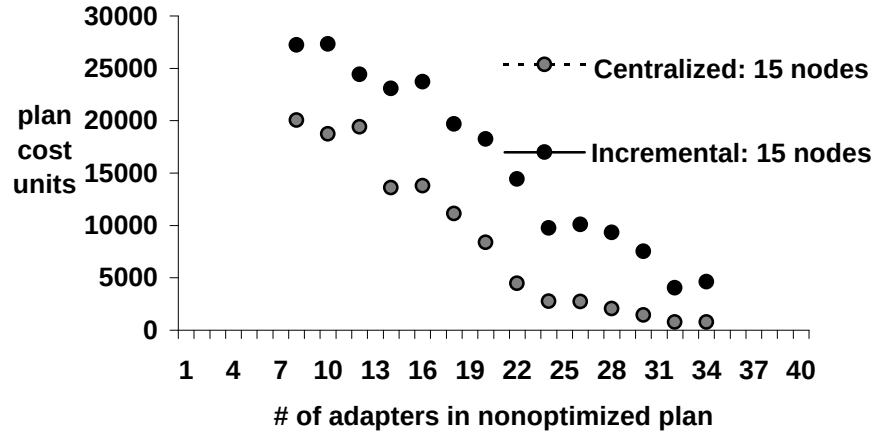


Figure 6.5: Incremental vs. central planning efficiency

6.3 Planning Process Test

6.3.1 The testbed

Computers and networks. The connection was tested with twisted pair sequential connections of up to four computers as shown on Figure 6.6. Dell Inspiron Omnibook 4150 laptops with 333 MHz processors were used for one set of tests and Hewlett Packard laptops with 500 MHz processors used for another set of tests; all machines used Linux Red Hat 7.0 with the 2.2.16 kernel. Xircom RealPort2 Ethernet 10/100 PCMCIA cards were used for the network connection between the machines. The source and destination machines run a user application and the Panda node concurrently. The priority of the user application was set lower on the source machine and higher on the destination machine to ensure the proper allocations of resources.

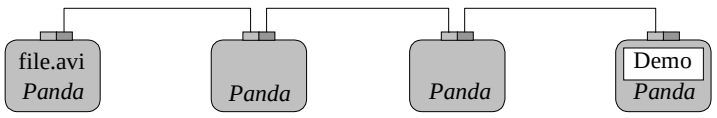


Figure 6.6: Panda peer-to-peer connection

Throughput of the network links is varied among 150 Kbps, 800 Kbps, 2000 Kbps, and 5000 Kbps using CBQ.

Adapters. We used two kinds of adaptations: *null adaptations* and real adaptations. Null adaptations do not perform any data processing; they are used to measure the overhead of just having an adapter in a connection. Filters and encryption were used as real adaptations. The filters drop particular packets with color or quality data and computationally are very economic; the encryption adapter performs heavyweight processing of the data.

The problem of synchronization. The following method was applied to measure one-way packet delivery. The packets were stamped with the local time on the source machine. Upon the arrival at the destination machine the stamped time was subtracted from the destination local time to obtain *measured time delivery*. The synchronization of the source and destination machines' clocks was done with NTP. The NTP server was located on the destination node. The source node synchronized itself to the destination local time before the first packet was sent to the destination. Then 20,000 packets were sent the destination. After the last packet was delivered, the source machine measured the *skewing value*. It was presumed that skewing grows uniformly by time. The *actual time delivery* was calculated with a formula for each data packet *n*:

$$ActualTimeDelivery(n) = measuredTimeDelivery(n) - \frac{skewingValue}{20{,}000} \bullet n$$

Applications. Three different applications using the UDP protocol were used for the performance tests. The latency of packet delivery and null-adaptations were tested on

a special application called *Connector* that was designed in Java for this purpose. Connector is able to generate data packets of different sizes. The average skewing value observed for 20,000 packets generated by Connector was 370 milliseconds.

The overhead of the planning protocol and real life adaptations were tested with the WaveVideo application [Frankhauser99], which generated a video stream using .avi files.

As an alternative to this video stream application, we used RAT (Robust Audio Tool), an audio stream generating application. It generated audio streams using .au files.

The quality of service was tested with the WaveVideo measurement package, which compared the initial data stream with the one that was actually delivered. The result is presented in PSNR units, which are the ratio of the initial stream to the error that occurred during the transmission.

6.3.2 Packet delivery and adaptation latency

Figure 6.7 presents packet delivery latency for different packet sizes. Panda without adaptations extends normal Internet latency three to four times, being a relatively slow Java application. Null adapters added to the connection make Panda overhead even heavier for packet delivery. The packet delivery latency also contains the adaptation latency. Error bars on this figure and all further figures show the value of standard error, unless otherwise indicated.

Figure 6.7 shows that adding Panda to a data stream increases its latency 50 to150%, with longer packets seeing less effect. Adding more Panda-enabled nodes or more adapters modestly increases the delay for each addition.

Figure 6.8 presents the latency of null adaptations. All adaptations were deployed on one of the nodes of the connection. Of course, without Panda no adapters can be deployed, so the extra latency for that case is defined as zero. Every Panda node always runs at least one *forward* adapter, whose only task is to forward a packet to a next node after all other adapters are executed. The number of forwarded adapters is equal to the number of connection nodes and is always present in a Panda connection, buts not counted on our graphs. Figure 6.8 shows that the overhead of a null-adapter is 2.4 milliseconds. A real adapter will take at least this value; actual data processing will cost extra latency.

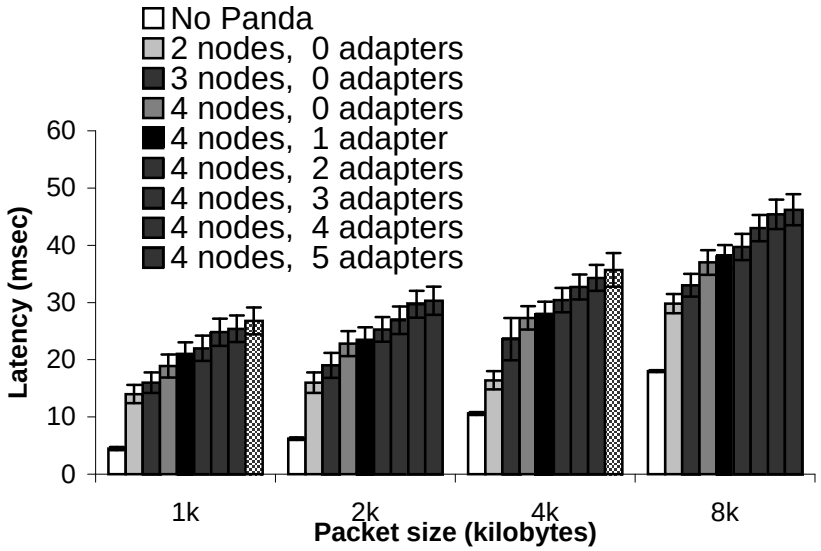


Figure 6.7: Packet delivery latency

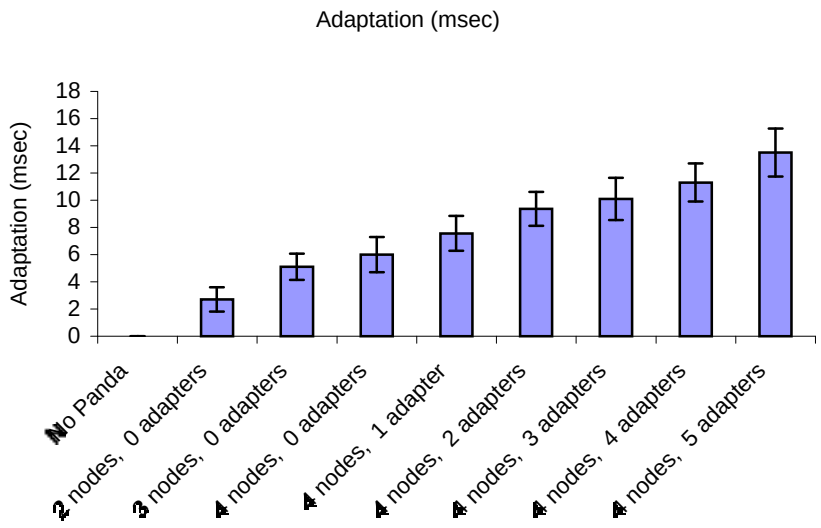


Figure 6.8: Adaptation latency

Figure 6.9 presents the packet loss that occurred in a data stream of 20,000 packets for different packet size. The data stream without Panda had no packet loss. No packets were lost for 2k-packet data stream was lost either. Packet loss increases with packet size because of extra memory allocation by the Panda Java code and associated with it extra latency. Figure 6.10 shows that Panda throughput grows with the packet size. At the same time Panda packet loss grows with the packet size, but it never reaches more than 0.5%. The packet size of multimedia applications varies anyway because some applications apply their own compressing protocols to the data packets. Error bars represent 95% confidence intervals.

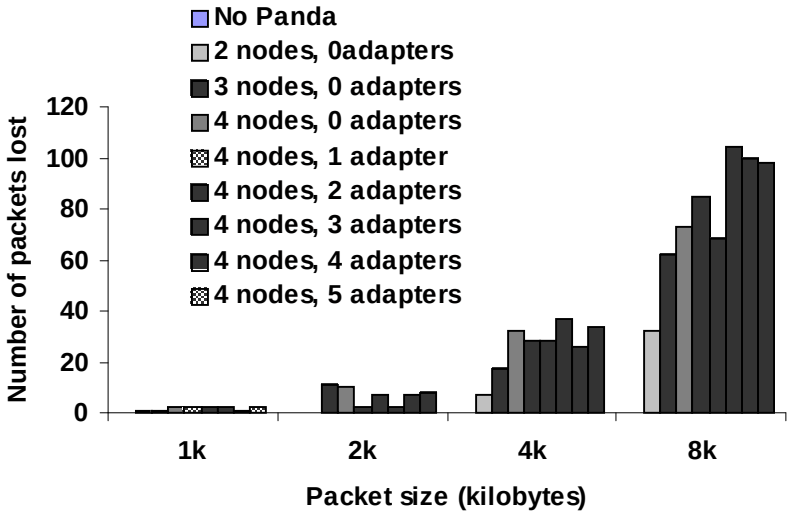


Figure 6.9: Packet loss

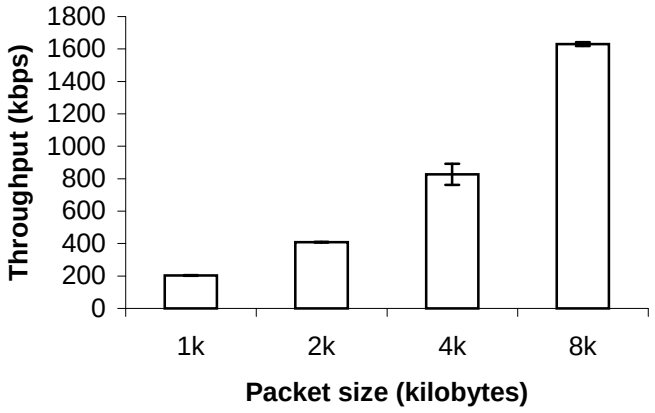


Figure 6.10: Throughput of Panda associated with packet size

Figure 6.11 presents the delivery time in milliseconds of an arbitrarily chosen 1000 packets. Axis X shows numbers of packets from 1 to 1000. The stream occurred on the connected four Panda nodes without adapters for 1k-size packets.

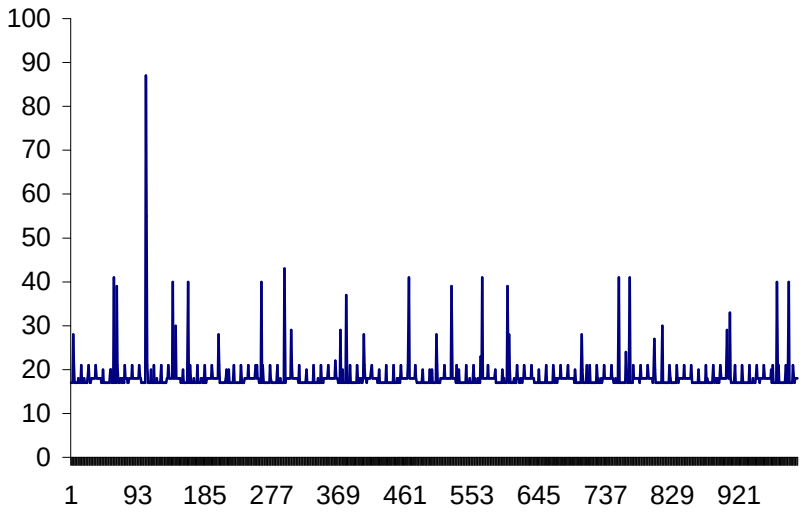


Figure 6.11: Sample of the distribution of packet delivery latency (axis Y) on packet numbers (axis X).

Figures 6.7 to 6.11 are obtained running the Connector application and null adapters.

6.3.3 Planning procedure latency with the connector application and null adapters

The planning procedure latency consists of planning data gathering latency, plan calculation latency, and plan deployment latency. Planning data gathering takes one round trip: the source node forwards the data gathering message to the end node and waits for its return. Planning data gathering for four Panda nodes takes 108 +/- 2.85 milliseconds.

For central planning, the time required to deploy the plan depends on whether the adaptations are pre-loaded on nodes. Obviously, if adaptations are pre-loaded the deployment latency is much shorter. Figure 6.12 presents the deployment latency for the case where adapters are not preloaded. The bars represent the deployment latency of 1 to

5 null adapters that were deployed on each of the connection nodes. The deployment on Node 1 is always fast because it is the source node, the storage site of all adaptations. In central planning, the more adaptations that must be transmitted to remote nodes, the longer the deployment process.

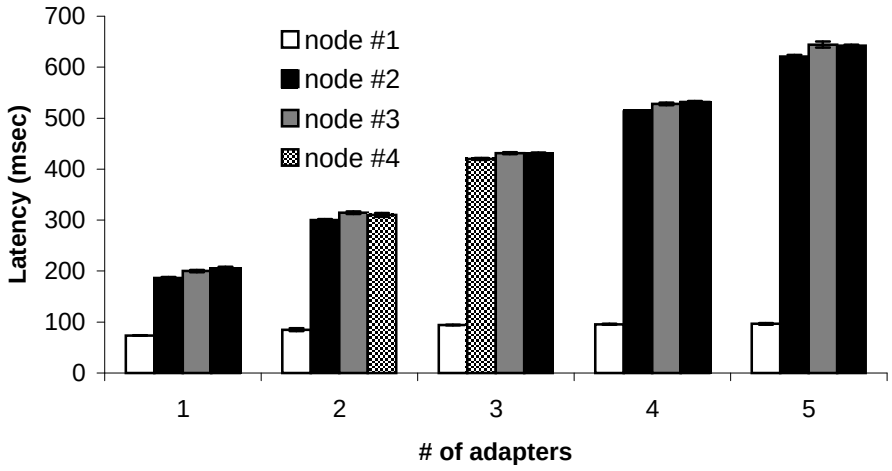


Figure 6.12: Deployment latency without pre-loaded adapters

Figure 6.13 presents deployment latency in the case of pre-loaded adapters. The latency of deployment is much shorter in this case because adapters need not be transmitted to remote nodes. However, the deployment protocol without adapter transmission still must instantiate the locally stored adapters, and that is why the

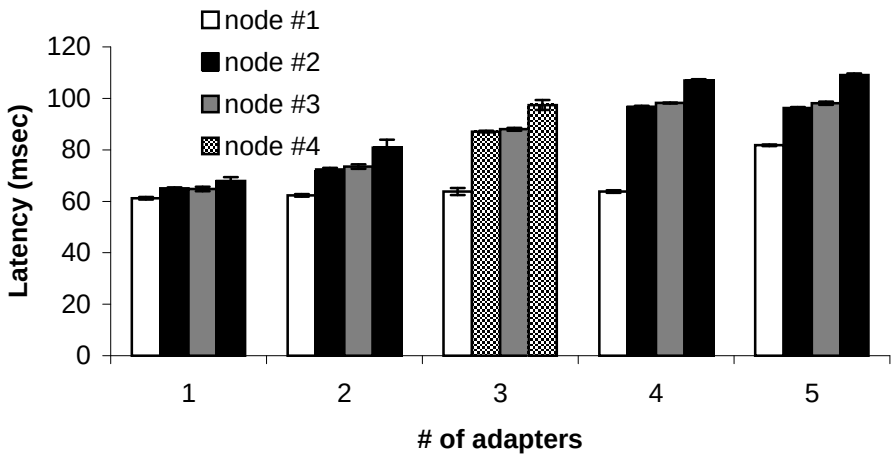


Figure 6.13: Deployment latency with pre-loaded adapters

deployment of more adapters takes longer per node.

Figure 6.14 presents the latency of the deployment protocol when no adapters are selected. In this case, the deployment protocol consists of querying messages sent by the source node to the intermediate nodes, asking them if they are ready to receive user data, and the acknowledgements from the intermediate nodes.

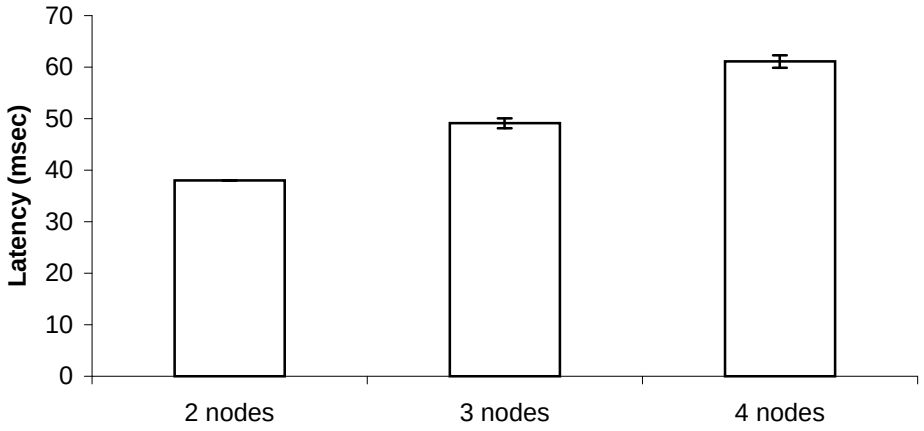


Figure 6.14: Latency of the deployment protocol without adaptations

Figure 6.15 presents the latency of the incremental planning process. The bars for deployment on the planning machine refer to adaptations that must be deployed on the machine that calculates the local plan. The bars for deployment on the next machine refer to the adapters that must be deployed on the machine downstream from the planning machine. The bars for deployment on both machines refer to cases where the incremental plan requires adapters on both of these machines. Incremental planning does not include any adaptation transmission. All nodes are presumed to be storing all adaptations that can be chosen by their local planners. The deployment of one adapter on either of machines has same latency. The deployment of one adapter on each of two machines takes slightly longer.

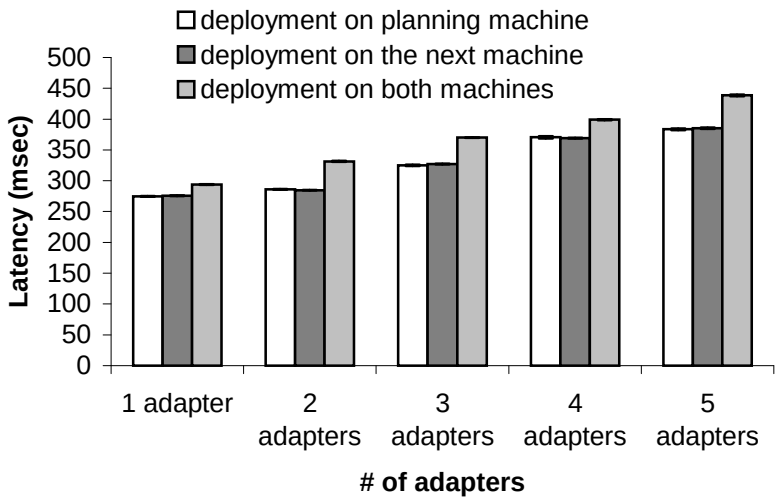


Figure 6.15: Incremental planning and deployment latency

Figure 6.16 presents the latency of performing both central and incremental planning. The bars marked as "Incremental" show the latency of the initial incremental plan. The bars marked as "Central" show the latency of the planning procedure if no incremental planning occurs. The bars marked as "Central plan with incremental plan in the background" show how incremental planning slows the central planning. Once the incremental plan is established at all nodes, data packets start to flow. These packets compete with the business of central planning, slowing that procedure down. Central planning, without incremental planning in the background, has a smaller latency because adapters can be transported to the node that will run them without competing with data packets for the node and link resources.

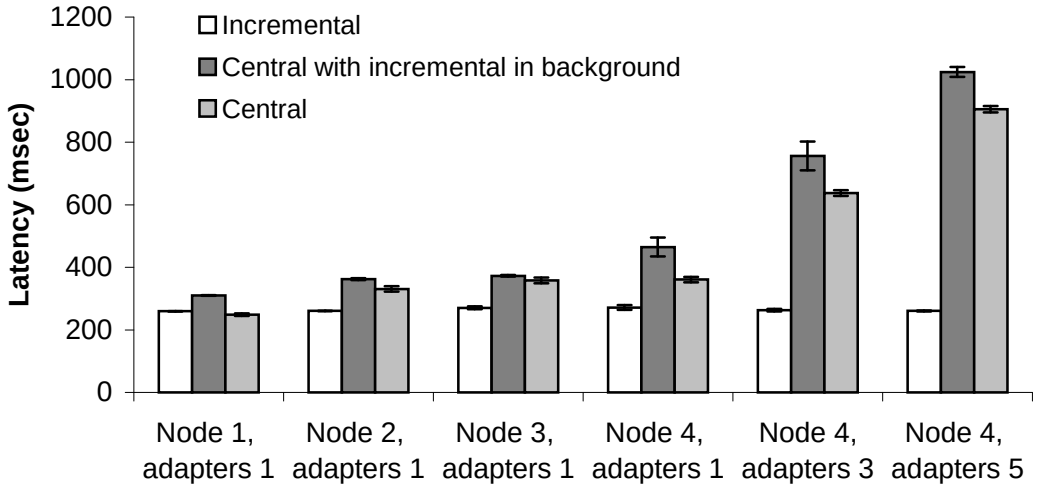


Figure 6.16: Incremental versus central planning latency

Figure 6.17 shows the number of packets that are forwarded under the incremental plan before the central plan is calculated and deployed. The bars marked "node 1, 2, 3 or 4 adapters 1" demonstrate the cases where central planning requires the deployment of one adapter on the 1st to the 4th nodes respectively. The further the adapter must be deployed from the source node, the longer the process of deployment; more packets are sent under the incremental plan. The bars marked "node 4, adapters 1, 3, 5" demonstrate the cases where the central plan requires the deployment of 1, 3 or 5 adapters on node 4. The more adapters deployed, the longer the deployment process; thus, more packets are sent under the incremental plan. The graph suggests that very short data streams, for example NTP, may not require central planning, as all of their messages will be delivered before the central plan is calculated and deployed.

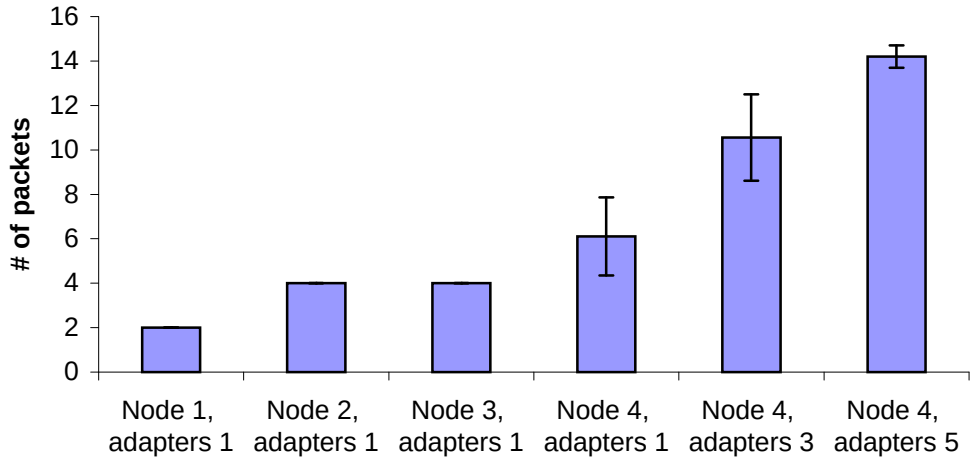


Figure 6.17: The number of packets sent under incremental plan before central plan is calculated and deployed

Sometimes the conditions of networks change after a connection is already established. If the changes are dramatic enough, the plan is no longer effective, and the system must replan. The process of replanning runs concurrently with the data packet stream and therefore takes longer than the initial planning. Figure 6.18 presents the latency of replanning. The white bars show the latency of the initial planning process, where one adapter is deployed on the third machine. The gray bars show the latency of replanning, where one adapter is deployed on the first, the second, the third, and the forth machines. Replanning takes at least 50% longer than the initial central planning. A replanning process, where an adapter is deployed on the source machine, still takes longer than the initial planning. This happens because the process of packet storing and forwarding on the source machine would occur in replanning takes more node resources than the process of packet storing only occurs in the initial centrlized planning. The

graph also shows that the further an adaptation is transmitted from the source node, the longer it will take to complete replanning.

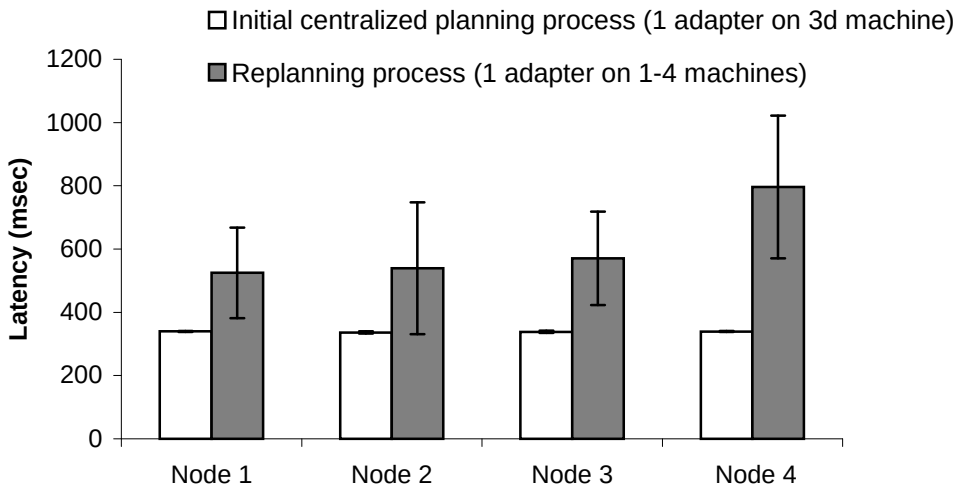


Figure 6.18: The latency of the replanning process

6.3.4 Planning procedure latency with real-life applications and adaptations

The results of the previous section were obtained by using the artificial Connector application and null adapters, and tests were conducted on Dell Inspiron machines. The following test results used the WaveVideo application and real adaptations, *ResolutionDrop* and *Encryption*, on Dell Inspiron and Hewlett Packard machines.

Figure 6.19 presents the centralized planning procedure latency for both the Connector and the WaveVideo applications. Of course the resolution drop adapter was not meaningful for the Connector data packets, but it was not an obstacle to use it for planning procedure measurements. The WaveVideo application generates data packets ten times as fast as the Connector application. This intensity puts an extra burden on the

CPU of the source node and suppresses Panda activity. Thus, the resource requirements of the user application influence the performance of Panda. Figures 6.19, 6.20 and 6.21 demonstrate the planning procedure latency, plan calculation latency, and deployment latency, respectively. Figure 6.20 shows that the plan calculation latency is strongly influenced by the requirements of the user application. Figure 6.21 shows that

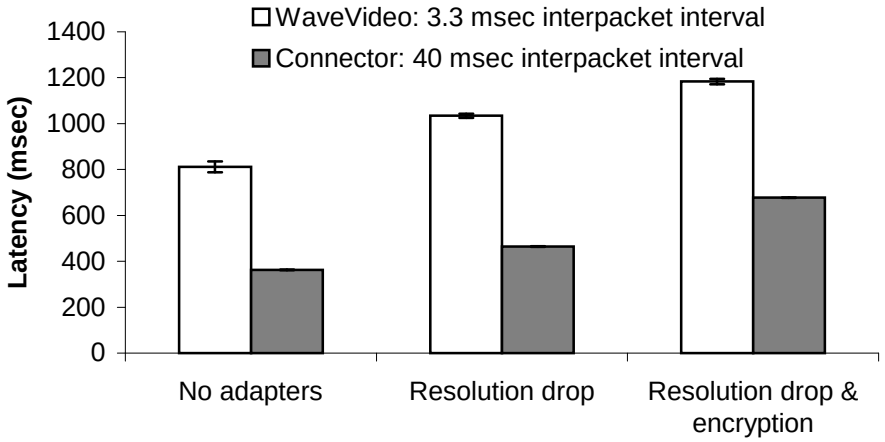
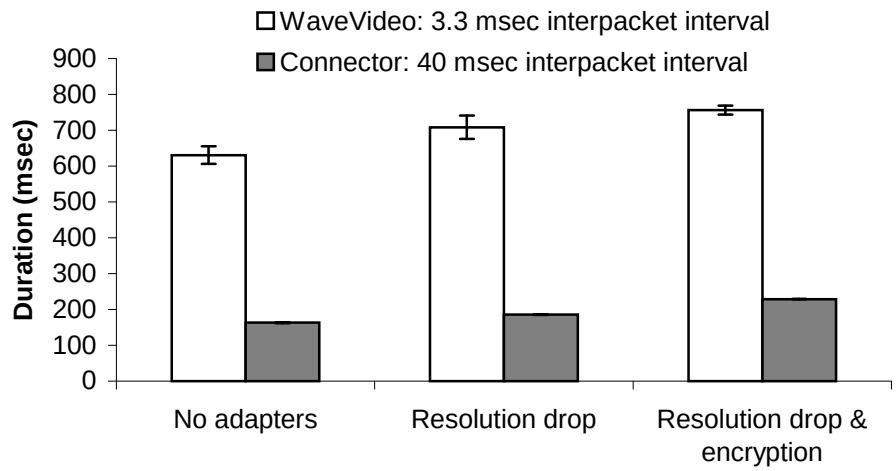


Figure 6.19: Planning procedure latency for the Connector and the WaveVideo applications

the deployment latency is almost unaffected, because most of the deployment activity takes place on connection nodes other than the source node.



.

Figure 6.20: Plan calculation latency for the Connector and the WaveVideo applications

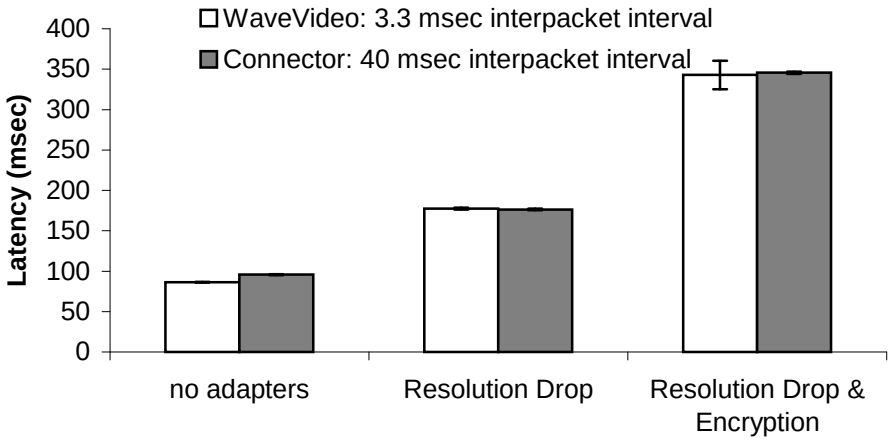


Figure 6.21: Deployment latency for the Connector and the WaveVideo applications

Figures 6.22 to 6.24 present the latencies for the planning procedure, plan calculation, and deployment, respectively, for the WaveVideo application for different network bandwidth that varies with different CBQ settings.

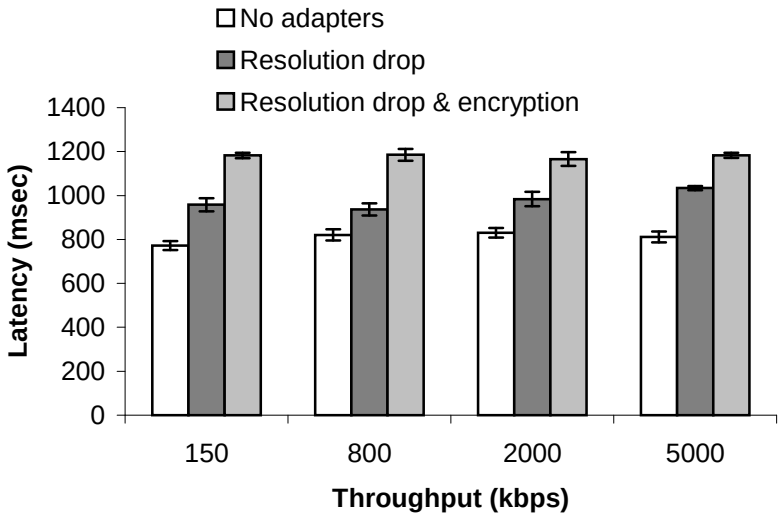


Figure 6.22: Planning procedure latency with Dell Inspirons

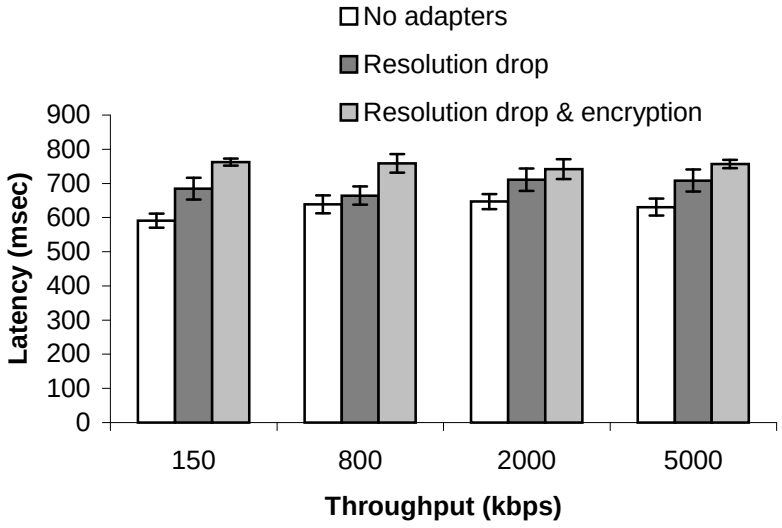


Figure 6.23: Plan calculation latency on Dell Inspirons

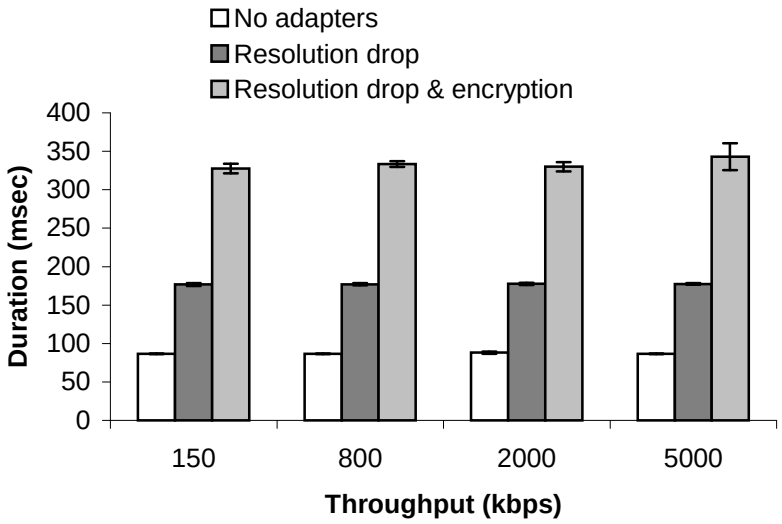


Figure 6.24: Deployment latency on Dell Inspirons

The graphs show little dependency of latencies for planning procedures on the network bandwidth, but a strong dependence on the number of adaptations.

Figure 6.25 presents the incremental planning latency for the WaveVideo application. The incremental planning procedure is faster than the centralized one (see Figure 6.22) even for a very small number of adapters (2 to 3) and four-node connections.

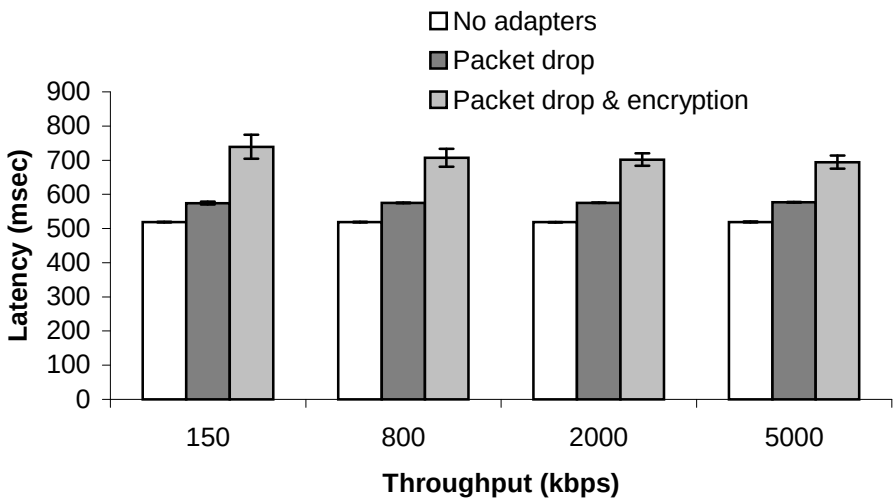


Figure 6.25: Incremental planning latency for Dell Inspirons

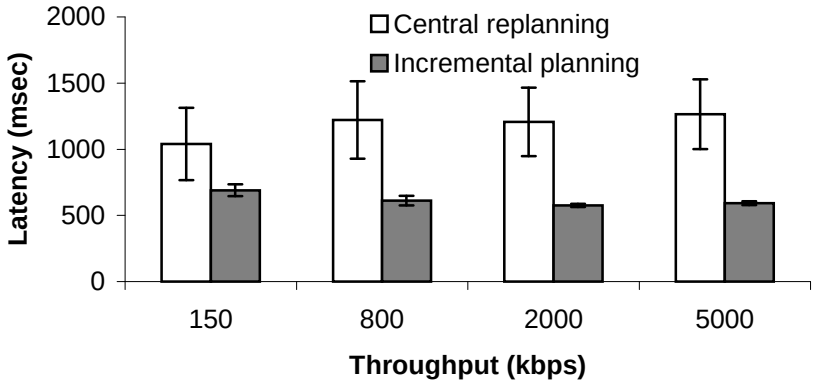


Figure 6.26: Incremental and central planning procedures (resolution drop only) on Dell Inspiron connection

Figure 6.26 presents the latency of replacing the incremental plan with the central plan. The gray bars on this figure are the same as the resolution drop bars on Figure 6.25.

The throughput has little effect on latency because the resolution drop adapter is very small, and can be quickly deployed regardless of the throughput. Incremental planning has a smaller latency than centralized planning.

Figure 6.27 presents the number of packets that were sent under the incremental plan before the central plan was calculated and deployed. The latency of the central planning procedure increases the number of packets. This figure again shows that very short sessions that transmit a small number of packets require incremental planning only. Earlier, in Figure 6.17, we showed the same number for the case with null adapters. The data in Figure 6.26 shows that heavier adapters increase the latency of centralized planning and thus increase the number of packets sent under the incremental plan.

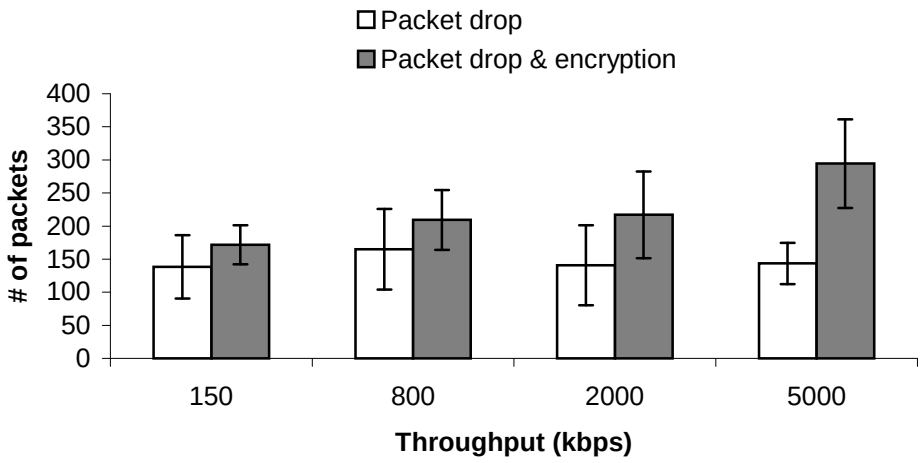


Figure 6.27: The number of packets sent under the incremental plan before the central plan is calculated and deployed

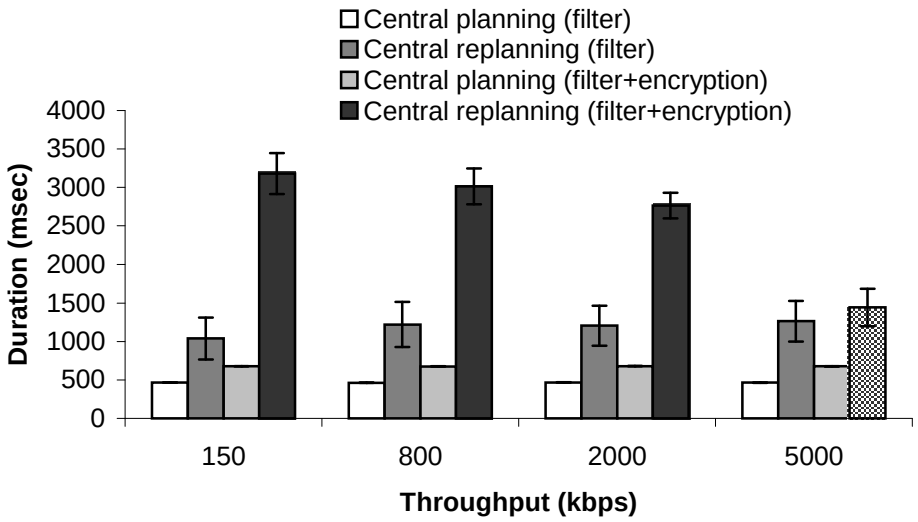


Figure 6.28: Central replanning process on Dell Inspirons

Figure 6.28 presents the tests of replanning during a session. Initial central planning represents the planning that occurred before the first packet has sent without performing incremental planning. Central re-planning occurs in the middle of the session concurrently with data packets. The encryption adapter is a relatively large piece of code and its deployment is seriously affected by the limited throughput of the connection; the re-planning procedure lasts from 1.5 seconds for 5000 Kbps to 3 seconds for 150 Kbps.

The test results with more powerful machines. The next series of experiments were run with more powerful HP Omnibooks (500 MHz of HP versus 333 MHz for the Dell Inspirons) to determine the effects of processor power on planning and adaptation.

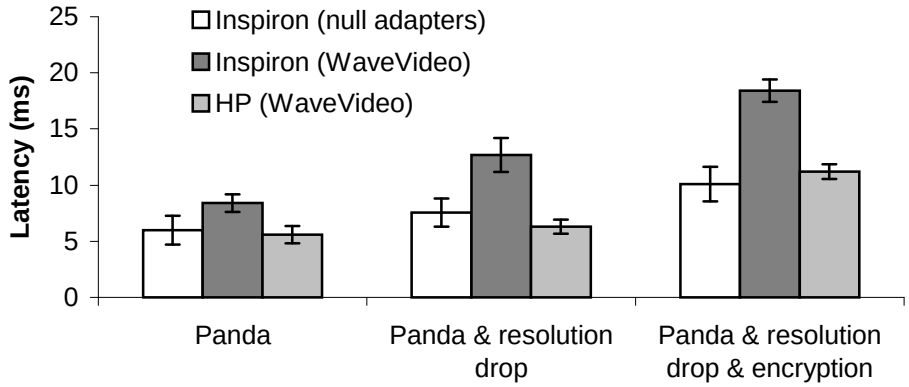


Figure 6.29: The comparison of adaptation latencies on Dell Inspiron and Hewlett Packard machines connections

Figure 6.29 presents the latency of the adaptation with real adapters on both the Dell and HP machines. Inspiron (null adapters) bars represent the adaptation latency with 0, 1 and 2 null adapters on Inspiron, which are compared to realistic cases. This figure shows that processing power has a major effect on the cost of running realistic adaptations, as would be expected.

Figures 6.30 to 6.32 compares the planning procedure, plan calculation, and the plan deployment latencies for Dell Inspiron and Hewlett Packard machines.

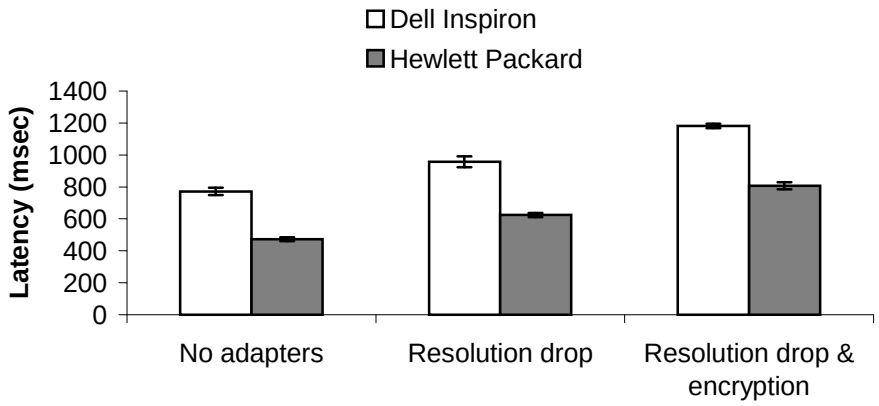


Figure 6.30: Planning procedure latency on Dell Inspirons and HPs

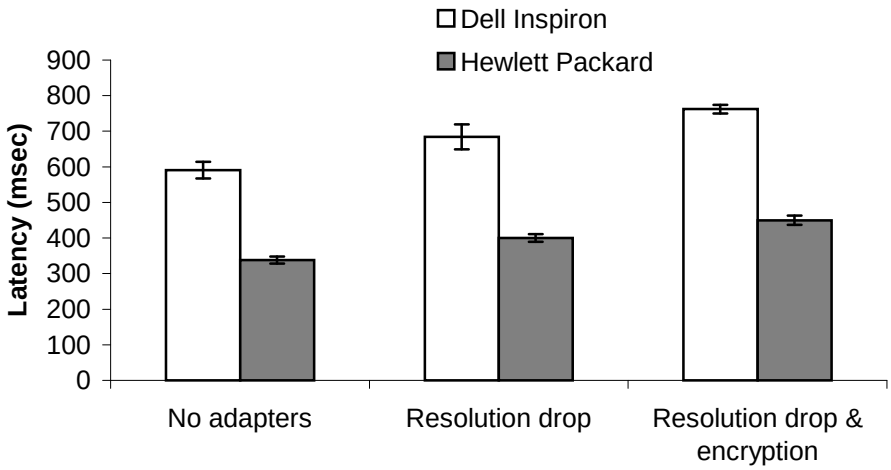


Figure 6.31: Plan calculation latency on Dell Inspirons and HPs

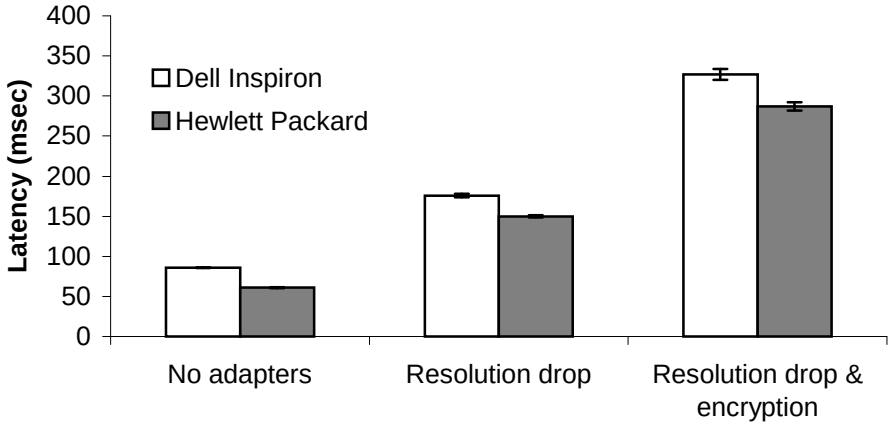


Figure 6.32: Deployment latency on Dell Inspirons and HPs

These figures show that planning is a CPU-intensive activity that can be assisted by more powerful machines. Much of the costs of deployment, however, are more dependant on the network than on CPU, so increasing CPU power provides less benefit in this stage.

The remaining tests were run only on Hewlett Packard machines. The planning data gathering procedure took 72 +/- 6 milliseconds for all situations. Figures 6.33 to 6.35 present the planning procedure, plan calculation, and plan deployment latencies.

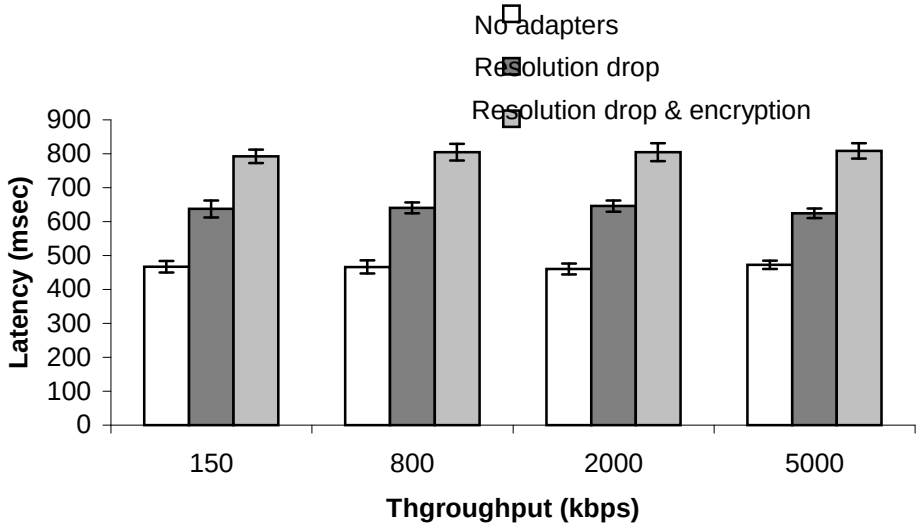


Figure 6.33: Planning procedure latency on HPs

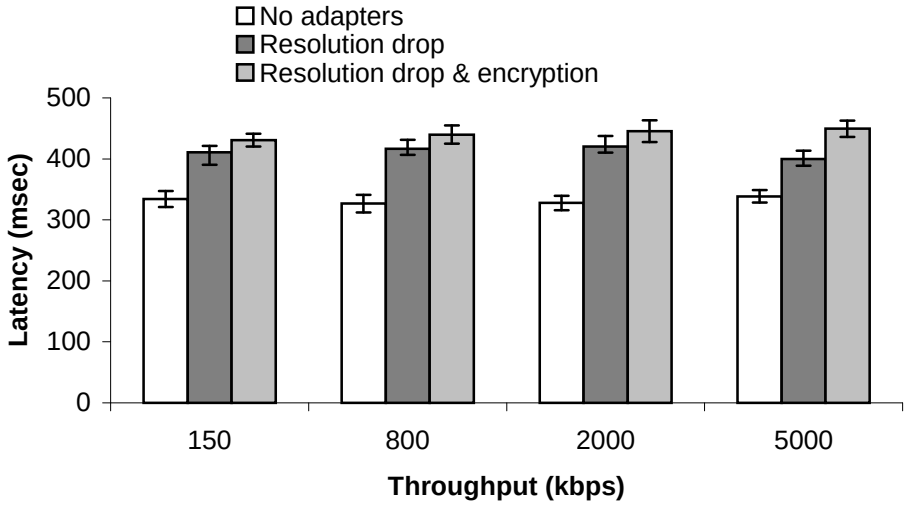


Figure 6.34: Plan calculation latency on HPs

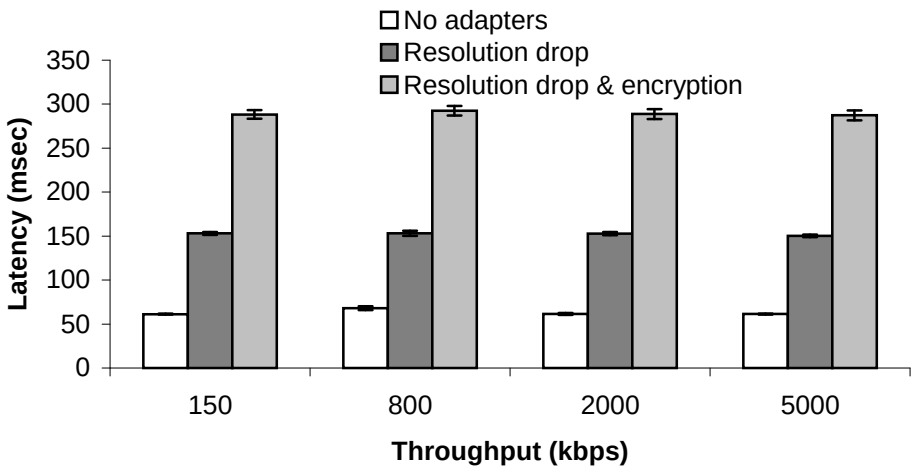


Figure 6.35: Deployment latency on HPs

Comparing these figures with Figures 6.21 to 6.22 obtained for Dell Inpirons, we can conclude that more powerful machines reduce the overhead of running the planning protocol and adaptations on Panda nodes. In both cases the planning procedure latency depends more on the number of adapters and less on the network bandwidth.

Figures 6.36 and 6.37 present the incremental planning, central planning, and replanning latencies for resolution drop and for resolution drop and encryption adaptations, respectively. The graphs show that incremental planning is faster than central planning, and central planning is faster than central replanning. The difference between initial central planning and central replanning is larger for larger adapters because the transmission of the adapters depends on the traffic between the connection nodes, and the replanning process competes with the data packet transmission. The bars for 150 kbps in Figure 6.37 show that the influence of the data packet traffic on that

difference is even more significant if the available network bandwidth is smaller, as would be expected.

Figure 6.38 presents the number of packets sent under the incremental plan before the central plan was calculated and deployed on the HPs.

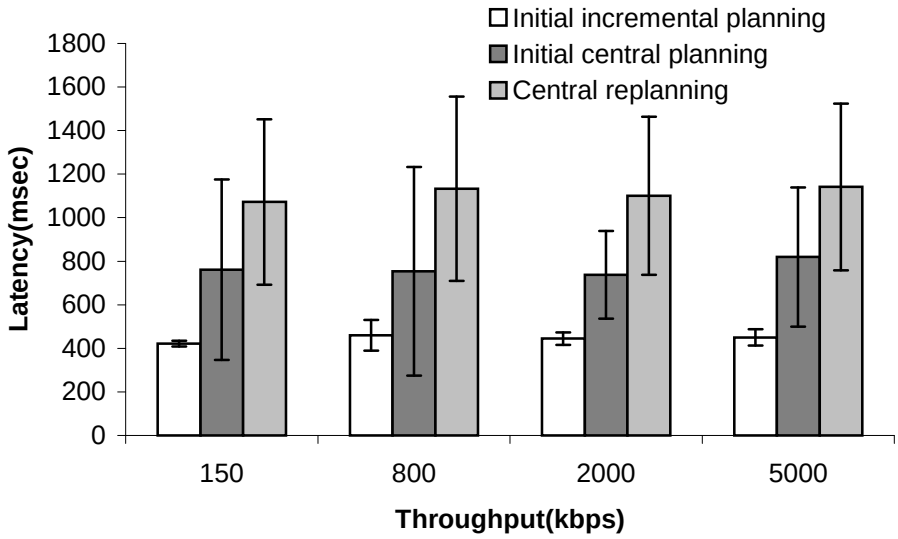


Figure 6.36: Incremental planning latency for resolution drop for HPs

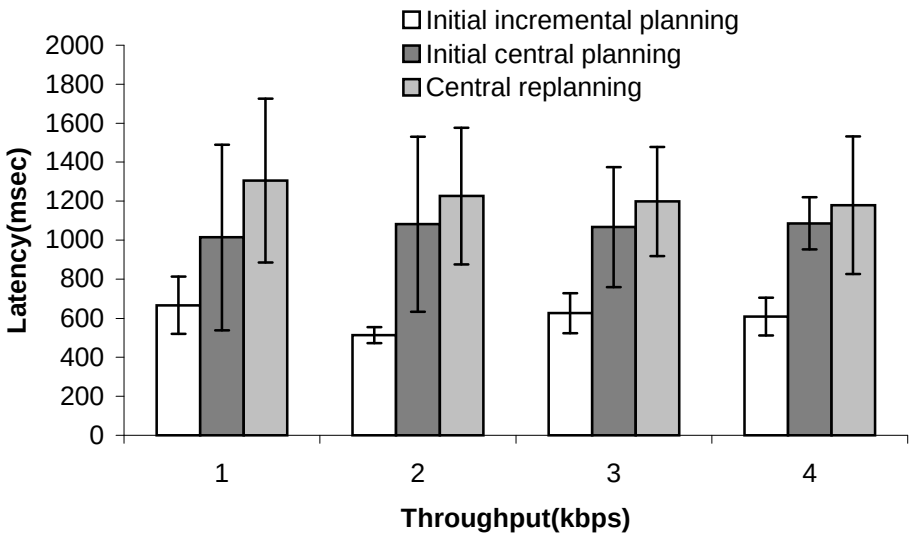


Figure 6.37: Planning latency for Resolution Drop and Encryption for HPs

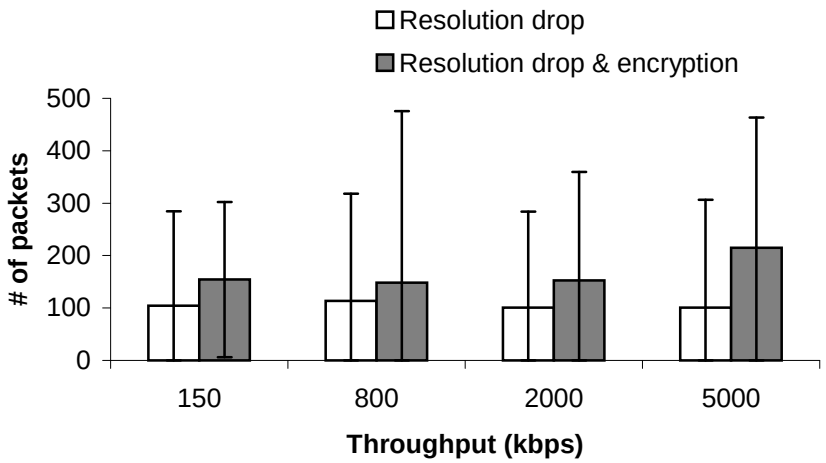


Figure 6.38: The number of packets sent under the incremental plan before the central plan was calculated and deployed

The number of packets sent under incremental plan before the central plan is calculated and deployed varies widely from 0 to 475. It makes the confidence intervals wide, allowing us to draw few conclusions about the effects of varying throughputs of different numbers of adapters.

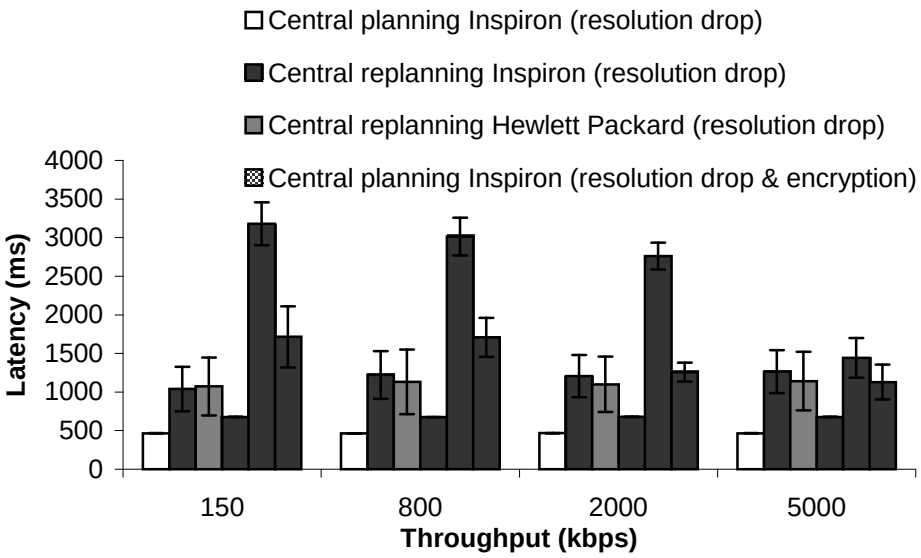


Figure 6.39: Replanning procedure latency on HPs and Inspiron

Figure 6.39 presents the replanning procedure latency compared to the correspondent initial central planning latency on Dell Inspirons and HPs. The graph shows that higher CPU power reduces the latency of the planning and replanning procedures. Slower machines also demonstrate a stronger dependancy on the available network bandwidth. Larger adapters make this dependancy even stronger.

The transfer of the resolution drop adaptation is not affected much by the throughput because it is a small adaptation. Encryption is a very large adaptation whose deployment takes much longer, and is more affected by competing data transfer traffic, thus varying from 1.5 seconds with 5000 Kbps throughput to more than 3 seconds with 150 Kbps throughput. Recall that the latency of the deployment that does not compete with data transfer traffic is presented on Figure 6.32. More powerful computers thus reduce the latency planning protocol and adaptation.

Another realistic application, RAT, was run to compare it to the WaveVideo application. Figure 6.40 presents the latencies of the plan calculation for WaveVideo and RAT. Both applications received plans calling for only the same encryption adapters.

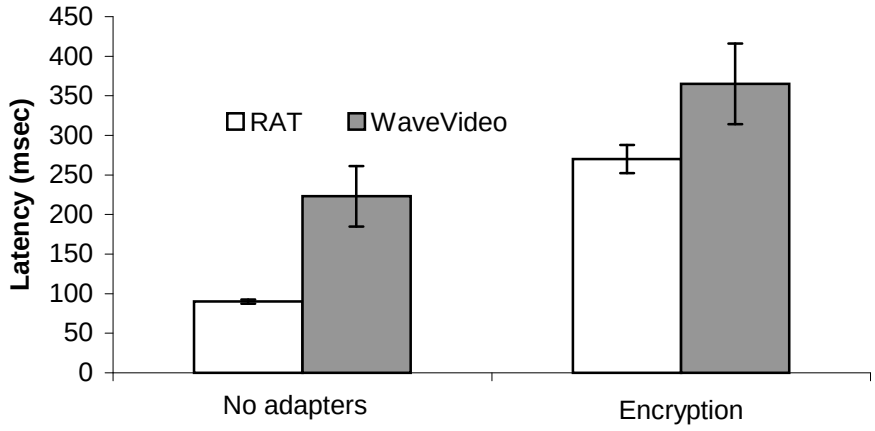


Figure 6.40: The plan calculation latency for Rat and WaveVideo applications

The RAT application transfers audio data, which is less intensive than video data. Since more resources of the source node can be used for the planning procedure, RAT receives its plan faster than WaveVideo.

6.3.5 Quality of service improvement

The Panda overheads described in the previous section are acceptable if Panda's adaptations improve application-meaningful quantities. Here we present evidence of such improvements. As we mentioned in Section 2.4, QoS is measured in dB of PSNR as conventional units. PSNR expresses the difference between sent and delivered signal.

Figures 6.41 and 6.42 present PSNR luminance and Cb values respectively for the WaveVideo application (discussed earlier) on Dell Inspiron machines with a link limited to 150 Kbps.

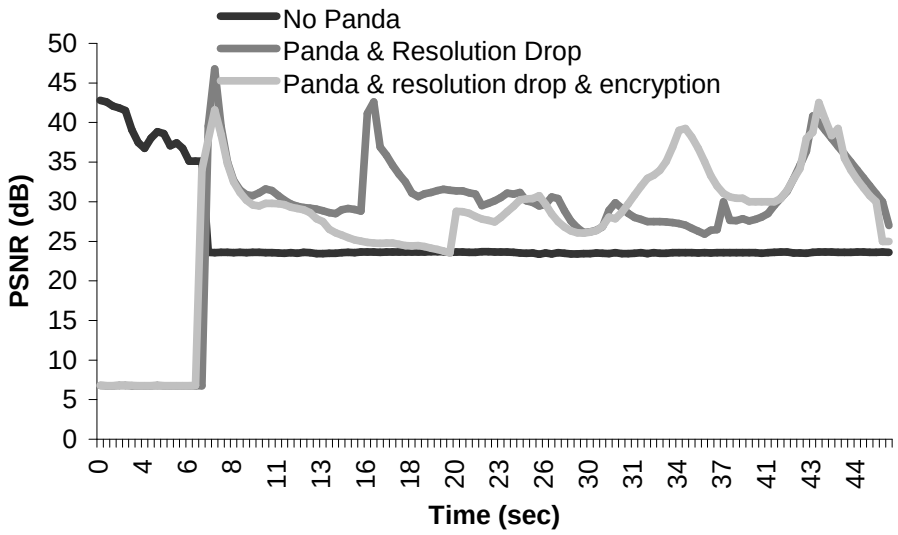


Figure 6.41: PSNR (luminance) on Dell Inspirons for 150 Kbps links

Without Panda, the curve falls once the channel is saturated; Panda's curve improves after its planning protocol is completed, providing better PSNR after around 20 frames. Panda achieves this improvement by dropping unimportant packets, thus allowing more important packets to arrive on time. The PSNR performance of Panda with resolution drop and encryption adaptation in some points can be even better than Panda with resolution drop only. One possible reason could be that Panda's extra buffering slows the data stream but reduces the undesired packet loss.

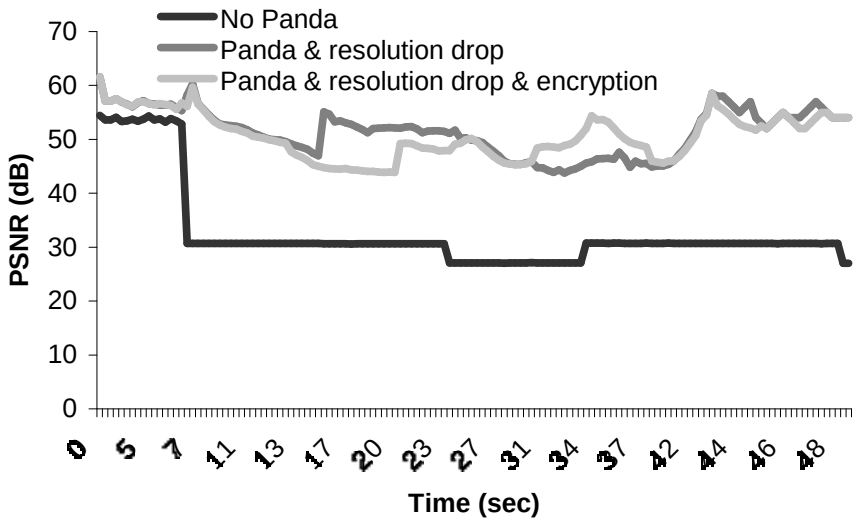


Figure 6.42: PSNR (Cb) on Dell Inspiron connection for 150 Kbps by video frames

Figures 6.43 and 6.44 present PSNR luminance and Cb values, respectively, on Dell Inspiron machines with 5000 Kbps links. In this case, Panda service is not necessary because the network is powerful enough to deliver packets on time. These figures demonstrate the importance of a network-aware planning process. If the resolution drop adapter were blindly applied or not applied without considering the network conditions, poorer PSNR would result for some cases.

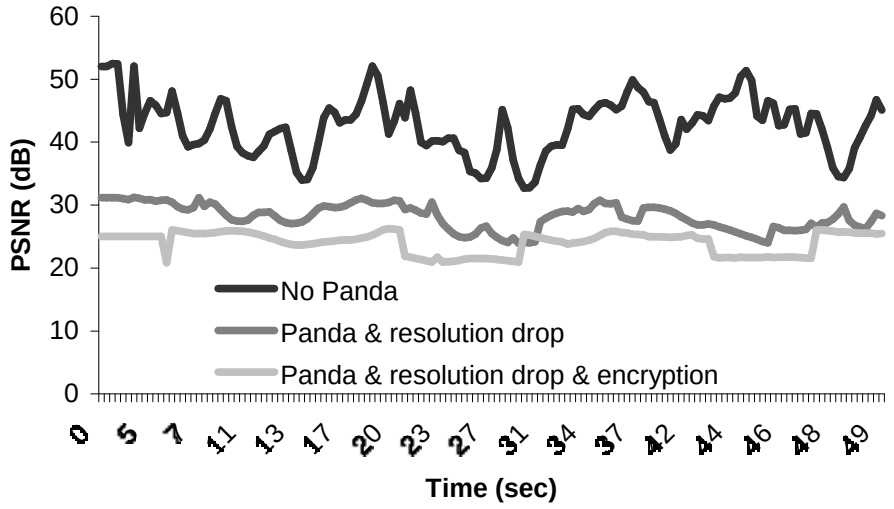


Figure 6.43: PSNR (luminance) on Dell Inspirons for 5000 kbps by video frames

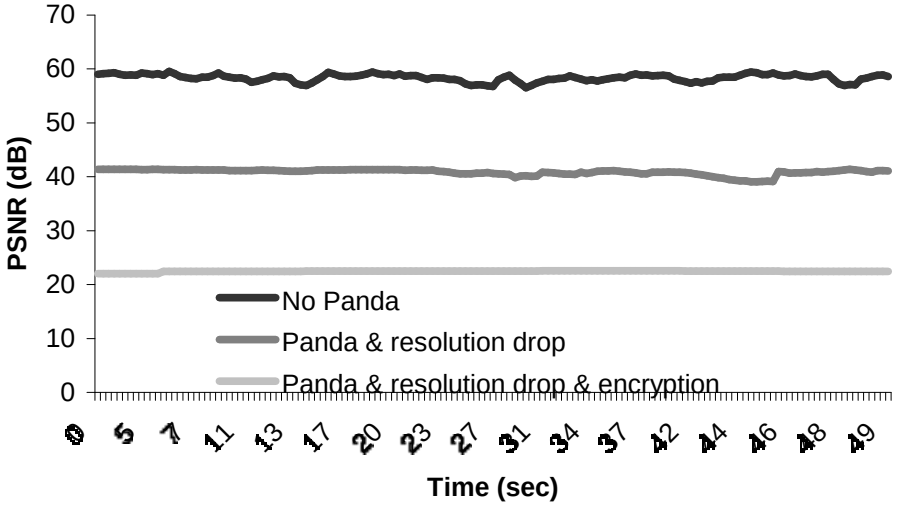


Figure 6.44: PSNR (Cb) on Dell Inspirons for 5000 kbps by video frames

More powerful machines can process more data packets and reduce packet loss in poor-condition networks, and thus increase PSNR. Figures 6.45

and 6.46 present PSNR values on Hewlett Packard machines. These figures show that Panda provides greater improvement with the more powerful machines.

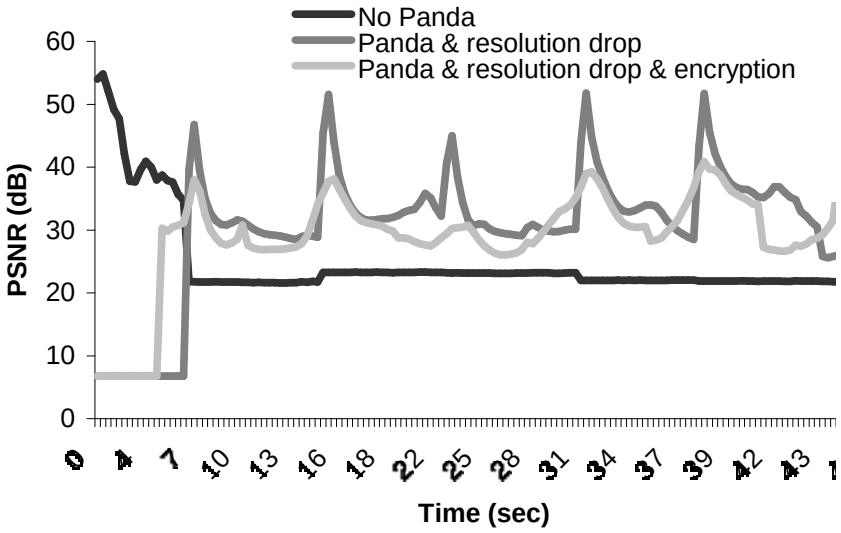


Figure 6.45: PSNR (luminance) on HPs for 150 kbps by video frames

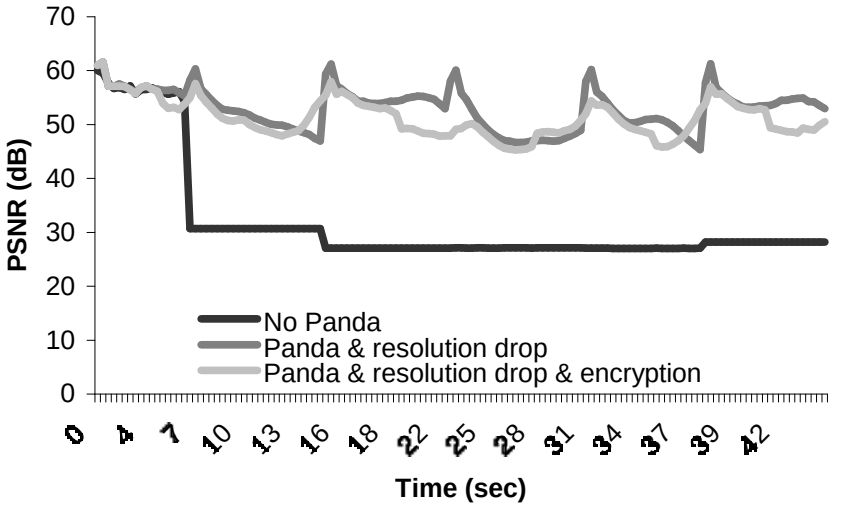


Figure 6.46: PSNR (Cb) on HPs for 150 kbps by video frames

Figures 6.47 and 6.48 present PSNR luminance and Cb respectively on Hewlett Packard machines with 5000 kbps. Even with this more capable network, in a few cases Panda provides better PSNR.

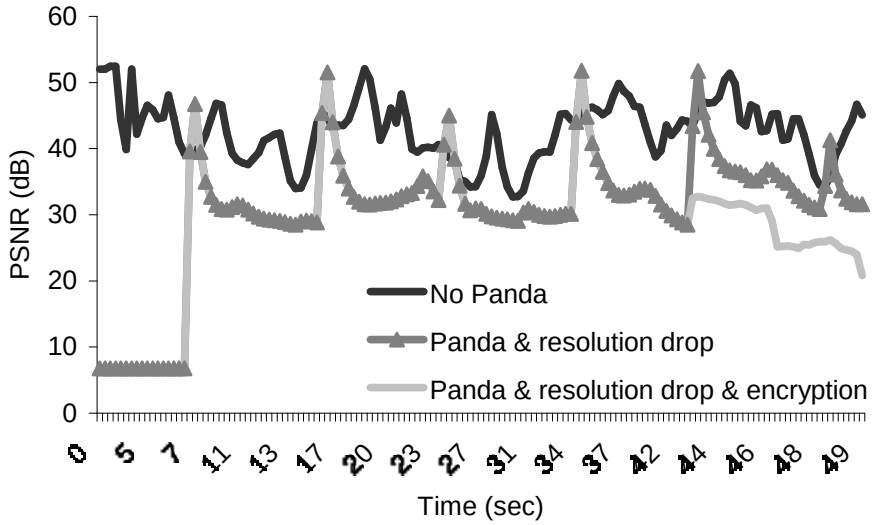


Figure 6.47: PSNR (luminance) on HPs for 5000 kbps links by video frames

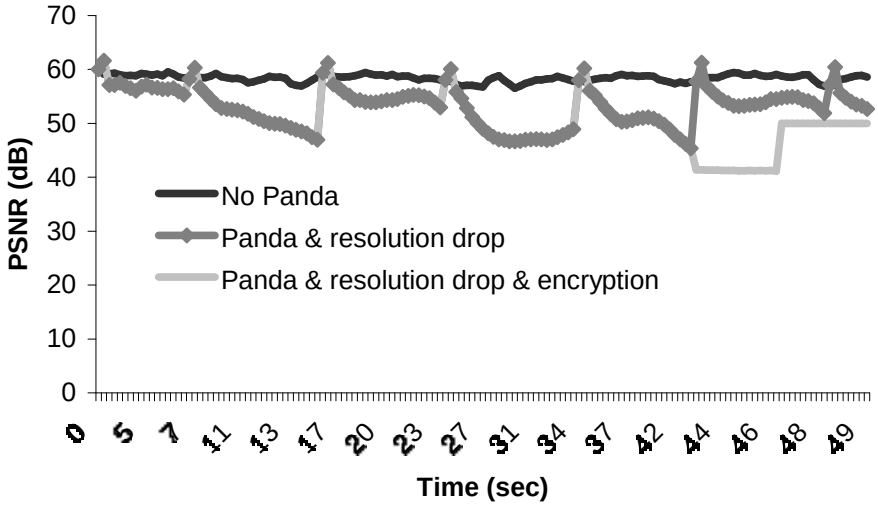


Figure 6.48: PSNR (Cb) on HPs for 5000 kbps links by video frames

In Figures 6.41 to 6.48, we include data for applying encryption along with packet dropping using Panda. For this data, Panda is providing a benefit beyond PSNR improvement by keeping the video secret. Without also dropping frames, however, much greater degradation in PSNR would accompany the improved security, as shown in Figure 6.49 on the Panda&Encryption bar. This data demonstrates the importance of considering all network conditions and possible remedies as a whole. It can be necessary to apply data compression just to compensate the effects of the Panda and its security remedies. Figure 6.49 presents PSNR values in various network conditions. This figure clearly shows that Panda with adaptations provides benefits for the networks with limited bandwidth.

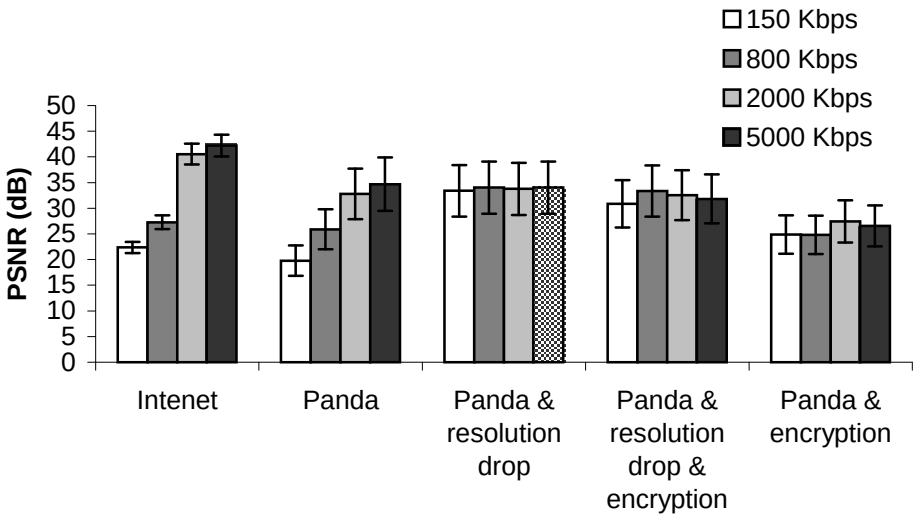


Figure 6.49: PSNR (luminance) on HPs

PSNR measurements can also be used for quantifying the quality of the calculated plans for video data streams. Consider the following example. Figure 6.50a shows an example of a connection. One link in this connection has poor bandwidth, which is insufficient to carry all the data. Another link is defined as insecure. If the link adjacent to the source requires encryption and the next link requires filtering, then the incremental plan will contain an encryptor on the source node and a decryptor and a filter on the next node (Figure 6.50b). It is clear that this plan is less optimal than the optimized plan that will put the filter and encryptor on the source node and a decryptor on the next node (Figure 6.50c). In the latter case, encryption and decryption will be applied to fewer data packets. Figure 6.51 demonstrates better PSNR for a filtered and then encrypted and decrypted data stream (the dark gray line) than with an encrypted, decrypted, and then filtered data stream (the light gray line). The black line shows the PSNR without using Panda. This example shows that a naive planner that allocates remedies next to links where problems occur can produce plans that are not only theoretically suboptimal, but that give poorer application-meaningful performance.

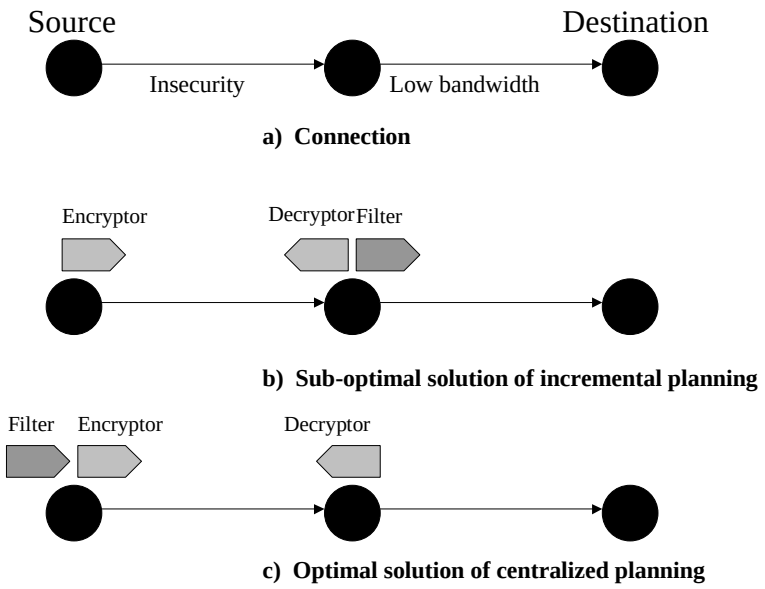


Figure 6.50: The advantage of central planning over incremental planning

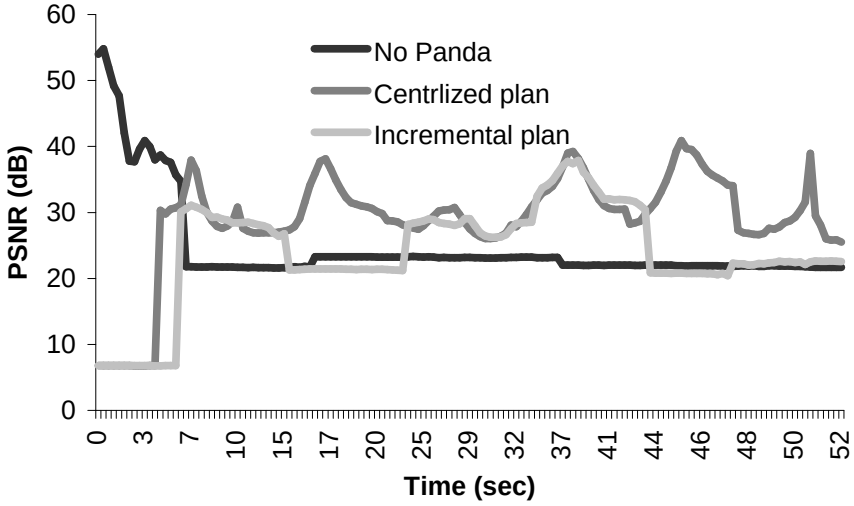


Figure 6.51: PSNR (luminance) for incremental and central plans

6.3.6 Discussion of performance

The tests show that the overhead of using Panda to adapt data streams can be compensated with a higher quality of service. The overheads are reasonable, particularly for relatively long-lived data streams. The latency added by the planning protocol is in the magnitude of 1 second. Panda also slows down the latency of data packets 4 to 10 times. The QoS however, can be improved up to 100%.

More computationally expensive user applications can increase the latency of plan calculation because application and planning processing compete on the source node. Plan calculation for a 10-times more intensive user application takes 4-times longer. This implies that the plan calculation should take place on the node that has more computational resources.

More powerful computers reduce the overhead of Panda and increase the delivered QoS. 1.6-times more powerful computers reduce 1.6-times the latency of the planning procedure and increase QoS by 30%.

Incremental planning can produce and deploy plans 50% faster than the corresponding central plan, but QoS for the incremental plan can be 45% worse in some cases, as shown on Figure 6.51. The number of packets that are sent under an incremental plan before a central plan is calculated and deployed varies from zero to some hundreds depending on the variance of the latency of the central planning procedure. Therefore, brief sessions should use incremental planning only. The length of the session should be explicitly indicated, for example in user preferences.

Replanning can take a number of times longer than initial planning because it runs concurrently with data traffic.