Adaptation Planning for Active Networks

Alexey Rudenko and Peter Reiher

UCLA, Computer Science Department
{arudenko, reiher}@cs.ucla.edu

Keywords: active networks, planning, heuristic search, real-time adaptation

Abstract. Active networks and adaptive middleware can improve network service through data adaptation and data rerouting. Automated distribution of adapters is desirable, because choosing proper adapters (from a large set) to handle arbitrary network conditions is complex. The paper presents an automated planning system that helps an active network system choose adapters and their locations. The paper describes the design and implementation of this system and provides performance data on its costs and benefits.

1 Introduction

Modern networks vary considerably in bandwidth, latency, jitter, reliability, etc. Wireless LANS, telephone lines, cellular telephony, and new devices lead to a wide variety of network conditions. One approach to handling such heterogeneity is to adapt data streams. If the sending and consuming applications can handle different forms of data, they can adapt transmissions to current conditions. However, while many applications have some flexibility about the type of data they consume, few are capable of negotiating proper versions of data streams with their providers.

An alternate solution is to use middleware to perform such adaptations. A simple form of such middleware is a proxy server that adapts data sent over wireless. More powerful alternatives like active networks allow adaptation at multiple points.

The adapters used by such middleware handle network problems using different methods to adapt user data. Some adapters handle limited bandwidth, e.g. compressors that run lossless compression algorithms or distillers that drop inessential portions of the data. Encryption adapters handle untrustworthy links. Caches and prefetchers can store user data at middleware adaptation sites. Wireless interface schedulers save power by frequently turning off their radios. Unreliable links can be improved by applying forward-error-correction (FEC) adaptation to user data.

Open network architectures (ONA) combine adapters to address multiple network problems, handling dynamic adapter deployment. This paper discusses planning in active networks, but the results can be applied to other ONA systems.

Choosing adapters and their locations is complex. The right adapters must be chosen in the right order, since misordered adapters can be counterproductive. For example, applying encryption before compression renders compression ineffective. If

multiple active network nodes are available, choosing which of them to host adapters is also difficult. Some choices will provide better efficiency and latency than others.

These problems are solved by creating a *plan* describing which adapters to run, in what order, and where. The problem is too complex for users, and too dynamic to be preprogrammed. This paper presents the design, implementation, and measured performance of an automated planning system that serves unicast connections by applying distributed adaptation to adaptation-unaware real-time applications.

2 The Planning Problem

Consider a user viewing confidential large images over several links. A wireless link is low bandwidth, insecure, and unreliable. An Internet link is insecure. The connection will benefit if compression, prioritization, encryption, and packet-level FEC [15] are applied to the wireless link and encryption applied to the Internet link.

Compression, encryption, and FEC have paired adapters. The first adapter adapts the data; the second adapter returns the data to its original state. In this example, someone must order and distribute nine adapters on three connection nodes. Exhaustive search of possible plans requires two seconds on a Dell Inspiron 333 MHz computer to examine 9600 plans, too long for many real-time applications.

In another example, field archeologists collect graphical images, video clips, and teleconferencing materials at an excavation area and exchange this data with remote colleagues [1]. Video data is sent via wireless to a nearby base station, connected to the Internet via DSL. This communication requires teleconferencing QoS. Confidentiality is important. Thus, the wireless link requires compression, encryption, and packet level FEC. The DSL link has lower bandwidth, requiring distilling and compression. The archeologists sacrifice video stream quality for teleconferencing real-time guarantees. The Internet link requires encryption. In this example, we must order and distribute eleven adapters across four nodes. Checking all of the hundreds of thousands of possible plans requires tens of minutes, an unacceptable latency.

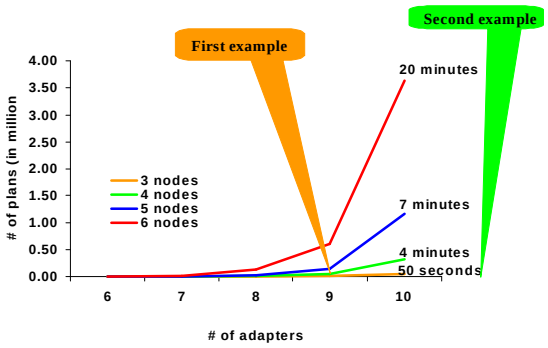

Fig. 1. Exhaustive search on a plan space

Figure 1 shows the size of plan spaces for differing numbers of connection nodes and adapters. Adapters are deployed in order, i.e. compressor upstream, decompressor

downstream, etc. The plan space grows exponentially with the number of nodes and adapters, making exhaustive search infeasible, even for realistic examples. With the growth of ad hoc networking, the number of nodes in a connection will grow (and with it, presumably the number of available ONA nodes), making exhaustive search planning even less suitable. The number of minutes shows the latency of exhaustive search for connections with 10 adapters.

One planning method is to use a small number of precalculated plan templates containing ordered dummy adapters assigned to virtualnodes. Real adapters and actual nodes are substituted at connection establishment. The planner chooses an appropriate plan template for current conditions, adds real adapters, and adjusts the template to the connection.

A simpler version puts all adapters on nodes adjacent to the problem links. The resulting plan may inefficiently use an adaptation multiple times. In Figure 2, compression is inefficiently applied twice on adjacent low-bandwidth links. Another simplification is to put all adapters on t he end nodes of a connection (Figure 3). Two adjacent links require compression and encryption. The end nodes should run both adaptations, but those nodes lack the resources to do so. Thus, the plan calculated with this method is unusable.

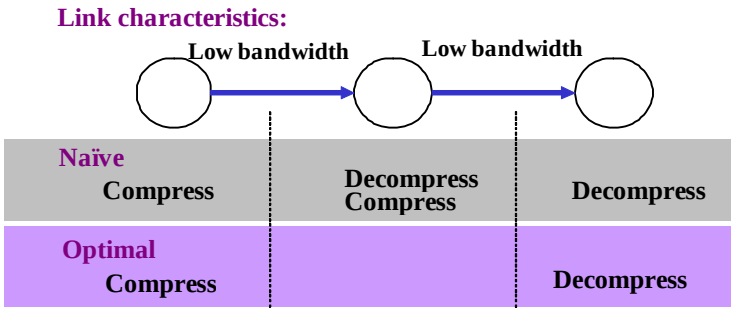

Fig. 2. Example 1 of simple template planning

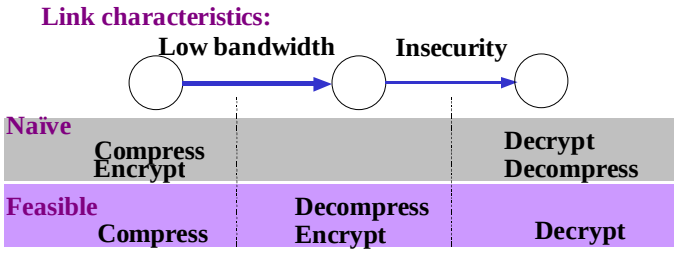

Fig. 3. Example 2 of simple template planning

Full-scale template planning is not flexible enough to handle the entire problem. The more templates the planner has, the more flexible a plan, but harder it gets to find

it. Assume N links in a connection and P problems on the connection links. Then 2^{NP} possible link situations can occur in a connection. Either an exponential number of templates must be provided, or some possible situations will not be handled properly. The number of possible adapter locations adds complexity. Node resource constraints may make a template unusable. The more complexities added, the less advantage one gets from the supposed simplicity of template planning.

This paper presents another approach that avoids the combinatoric explosion of the planning problem. In this approach, the planner calculates a plan online during the connection establishment using a heuristic search in the plan space.

3 The Planner Design

Our planner had to meet several requirements. It had to be fast, to serve real-time applications. Its plans had to use adapters consistently so that no adapter inhibits the work of another adapter or destroys the semantics of user data. The planning system had to be extensible, allowing later addition of new adapters.

The planner uses *adapter descriptions*, which contain the following data:

Problem ID. Network problems are described with unique problem identifications, e.g. low bandwidth, insecurity, etc. Problem ID is used to match observed problems to suitable adapters.

Method of resolution. A given problem can be solved in more than one way. For example, the low-bandwidth problem can be solved by compression, distilling, prioritization, etc.

Adaptation effect. The properties of applying an adapter, such as estimated efficiency of adaptation. The planner uses this adapter efficiency attribute to choose between two adapters that solve the same problem. The data size impact describes the adapter's effect on data throughput, and whether it is lossy. A compressor reduces the amount of data sent, while FEC increases it. The planner extends the effect of the adapters that reduce data volume over more connection links. The planner prefers lossless adaptation to lossy adapter.

Adaptation cost. Adapters have costs: CPU, memory, etc. Adapters use can also have monetary or deployment latency costs. The planner will try to minimize these costs.

Preconditions of adapter use. Some adapters require that certain pre-conditions be satisfied, such as data must be in a particular format. Once a Lempel Ziv compressor or encryption is applied, another Lempel Ziv compression becomes useless. We use various compressabilities as preconditions that characterize whether a particular compressor can be applied to user data.

Postconditions of adapter use. Adapter postconditions describe the properties of the data after adapter application, allowing them to be matched to the next adapter's pre-conditions.

The planner requires some information to calculate a plan. Planning data consists of stream characteristics, stream requirements, user preferences, link conditions, and node resources.

Stream characteristics. These describe the user data stream, such as format of data and whether the data is compressed or encrypted by the user application that generated the stream.

Stream requirements. These show the minimal useful throughput of the stream, confidentiality requirements, etc.

User preferences. Users can optionally choose a particular problem solution (e.g., dropping video stream resolution instead of color), or require the use of a particular adapter.

Link conditions. Link conditions affect the data transfer. The planner compares stream conditions, stream requirements, and link conditions to detect problems and select remedies. Link conditions include bandwidth, jitter, latency, reliability, security, etc. Small fluctuations in a link attribute can be handled by choosing adapters able to handle a wide range of conditions.

Node resources. A plan's use of node resources must be verified as feasible during plan calculation. An ONA node's policy on resource sharing (priority, client quotas, purchased levels of service) affects feasibility, preventing one connection from hogging all node resources. Node resources consist of CPU, hard drive, memory, associated costs, etc.

The planner uses heuristic search in the plan space to calculate a plan, executing three consecutively executed processes: *adapter selection*, *adapter ordering*, and *plan optimization*.

Adapter selection allows the planner to verify plan feasibility at any point, since it is working with real adapters. But if the wrong adapter is selected, it will affect all the following planning steps, and changing the adapter requires starting the entire process over again.

Adapter ordering uses precalculated templates. These are used exclusively to determine acceptable ordering of adapters, avoiding the earlier -mentioned problems of template planning.

After adapters are selected and ordered, the plan is optimized.

Dividing planning into these steps reduces the search space. Some early choices affect further steps, which could prevent finding the best plan. The planner does not guarantee optimality. The selection of effective heuristics is the key issue for the heuristic search. We verify our heuristic choices through comparison with exhaustive search.

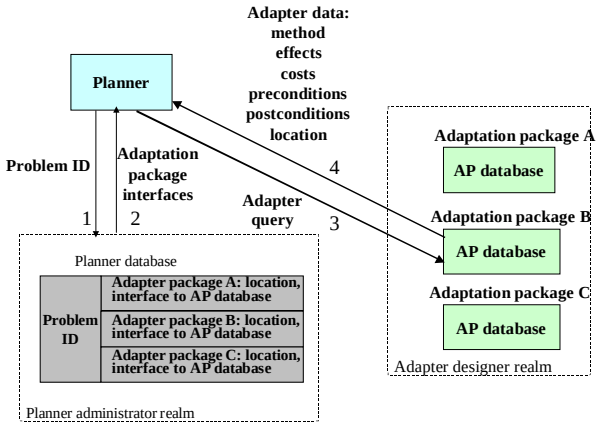

Fig. 4. Adapter selection

Adapter selection. Adapter selection is the first step of planning (Figure 4). The planner analyzes stream characteristics, stream requirements, and link resources to detect problems, and uses problem IDs to select the adapters. The planner searches its database by problem ID, step 1. Adaptation package records (APRs) are associated with a problem ID, which also contains the package's location and an interface to the package's own database. More than one APR can be returned to the planner for a single problem ID, step 2. The planner uses various techniques to choose adaptation packages, such as self-learning or always choosing a preferred package.

The planner uses the chosen APR's interface to query the adaptation package database, which returns the real adapter data step 3 and 4. The adaptation package database is provided by the adapter designer. This two-level access to adapter data allows the planner and adaptation designers to work independently, relying on the interface to reconcile their requirements. This process produces an unordered set of adapters associated with nodes adjacent to the problematic links.

Adapter ordering. Partial-order plan templates are calculated off-line. The adapters in the templates are represented by adapter methods that def ine proper adapter order (e.g., Figure 5). Some adapters are arranged sequentially, such as distilling, compression, encryption, and FEC. Other adapters are arranged in parallel, such as format conversion, data storage, and filtering; their order is determined later, as described below. A format converter can be applied to data before or after distilling, as long as the data format is recognizable. Caching should be applied before FEC, to avoid wasting caching resources. Format-insensitive filtering is not limited by order.

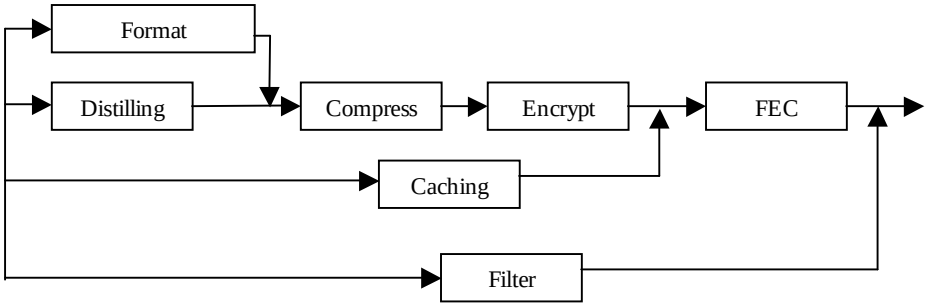

Fig. 5. An example of partial-order template plan for adapter ordering

The planner then replaces adapter methods with real adapters. Parallel orderings are resolved using additional constraints of the real environment, such as application requirements, adapter pre- and postconditions, and network conditions.

Adapter conflicts are detected by analyzing their pre- and postconditions. The planner resolves conflicts by changing the adapter order if possible, or it adds a format conversion adapter to resolve the conflict. If a conflict cannot be resolved, the planning process must restart with adapter selection.

Adapter selection and ordering produces a chain of local plans where every adapter is selected, ordered, and located in the nodes adjacent to the link with the problem. Figure 6 shows an example from the archeology scenario. This plan can be used, but it may be inefficient. It may include redundant adapters (see Section II). Or link resources can be wasted when compression is applied to one link and not to others.

Plan optimization. Plan optimization uses a variant of a recursive best -first search algorithm. The initial point is the chain of local plans described above. The optimization algorithm applies transformations to the current point, evaluating each transformation's value and feasibility. If a transformation produces a better value of the evaluation function and the new plan is feasible, the plan is recorded as a potential solution. The goal is to find the minimum of the evaluation function. The transformations are based on merging neighboring plans, preserving the order of adapters from both plans. Merging two plans involves three nodes: the node common to both plans (the median point) and two other end nodes. Merging moves adapters from the median point to the end nodes. After all adapters are moved, redundant adapters are dropped, resulting in a new plan to be merged with its neighboring plan. The goal is to merge as many plans as possible–ideally all plans of the connection.

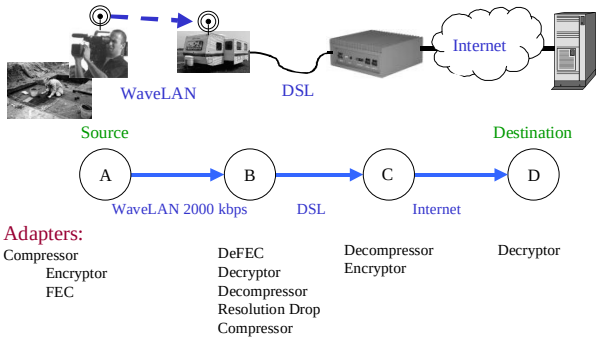

Fig. 6. The result of adapter selection and ordering: the chain of local plans

Plan merging can be interrupted for various reasons, including a wor se evaluation function value, insufficient node resources, or unrecoverable adapter conflict. If merging is interrupted in one place, it can be resumed in another place in the chain of plans. If time runs out, the merging process stops and the best plan found so far is chosen. An example of creating an optimal plan by merging for our archeological scenario is presented in Figures 7 and 8. First the plans on links AB and BC are merged, then the resulting merged plan is merged with the CD link plan.

We use an evaluation function to drive the optimization process:

$$f = \sum_{links}^{link\ resources} \sum_{k} \alpha_k lr_k + \sum_{nodes}^{node\ resources} \sum_{m} \beta_m nr_m ,$$

where lr and nr are link and node resources, and α and β are normalizing weight coefficients. We optimize the plan by minimizing the link and node resources usedyb the connection. Although the function does not contain latency explicitly, dropping the redundant adapters reduces adaptation latency.

Feasibility verification involves comparing the resources required by the adapters to available node resources at the locations where these adapters must be executed.

Thus, the planning algorithm contains the following steps:

1. Detect network problems by analyzing the application stream characteristics and requirements and available link resourc es.

2. Select adapters that solve any detected problems.

3. Order the chosen adapters on the nodes adjacent to problematic links using the partial-order plan template.

4. Merge the first two plans from the resulting chain of plans.

5. Evaluate the result with the evaluation function.

6. Verify the consistency of adapters.

7. Verify the feasibility of the resulting plan.

8. If the plan is better according to the optimization function, if the adaptations are consistent, and if the plan is feasible, record it.

9. Merge the plan that is the result of the previous merging with the next-neighbor plan. If all plans are merged, return the last recorded plan and stop.

10. Go to step 5.

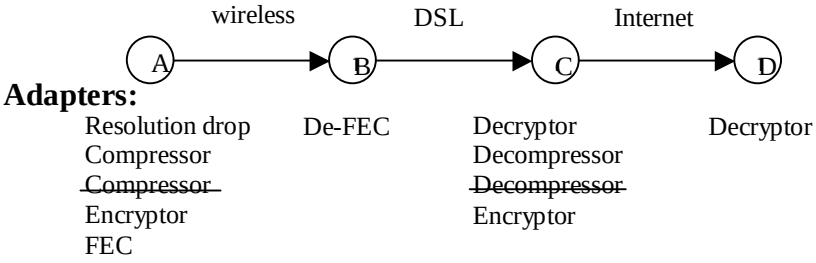

Fig. 7. Merging plans AB and BC in archeology example

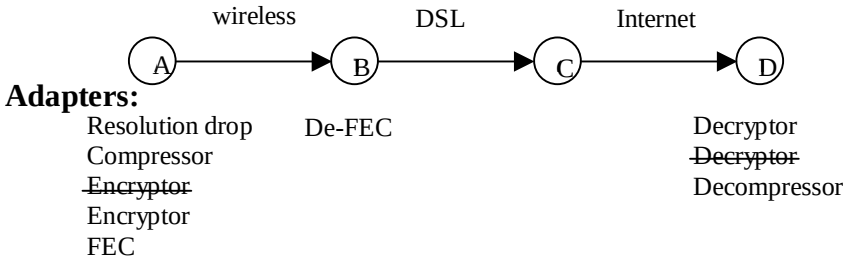

Fig. 8. Merging plans AC and CD in archeology example

The result of adapter selection and ordering is a plan that can be used as a solution. The transformations change the state of one potential solution to another potential solution. The minimization of the ev aluation function helps to find the best solution, but it may only find a local minimum. The optimization produces a best-effort plan.

During the merging process we preserve the original order of adapters. However, the consistency of adapters may be v iolated. Thus, we must verify adapter consistency during plan merging. If it is violated, it must be restored using the techniques described earlier. If adapter consistency cannot be restored, plan merging fails.

With p adapters and n nodes, the total merging complexity is $O(pn)$. Each optimization step has complexity $O(n)$. Hence, the total complexity is $O(pn^2)$.

4 The Planner Implementation

The planner was implemented in the Panda active network middleware [5]. Panda uses active network technology to serve adaptation-unaware applications. Panda intercepts normal data packets set to a destination and converts them into active packets, which are stored on the source node until a plan is calculated and deployed. Panda collects the planning data, invokes plan calculation using the planner described in Section III, and deploys the plan. The source node then sends the active version of the application packets to the destination. They are intercepted at any Panda nodes along the path that have had adapters deployed for this connection.

This planner can calculate a local plan for a particular connection link or a whole-connection centralized plan for all links. In this implementation, local plans are calculated as a fast, but often subotimal, solution to sta rt data transfer quickly. Local plan calculation involves only adapter selection and ordering. A whole connection plan is simultaneously calculated and ultimately replaces the local plan.

If Panda detects that the network conditions change sufficiently during a connection that the existing plan cannot satisfy the connection requirements, Panda replans, substituting a newer, more suitable plan for the old obsolete plan. The unused adapters are garbage collected by Panda nodes later.

5 Performance

In this section we present some performance measurements of the planner and the Panda planning implementation described above; more results are presented in [16].

We tested a Java implementation of the planner on 333 MHz Dell Inspiron laptops. Connections were generated in a randomized 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 LZ compression. Each link was designated secure or insecure, requiring no adaptation or encryption and decryption, respectively. We generated a resource availability for each node, expressed as the number of adapters the node could run.

Figure 9 shows the latency ratio between heuristic and exhaustive search for differing numbers of connection nodes and adapters. For small cases (e.g., three nodes, 4 adapters), exhaustive search is better, but the planning latency is less than 10 milliseconds here. For larger cases, heuristic search was better, and had larger latencies. Exhaustive search becomes infeasible as the number of nodes and adapters grows. A hybrid approach could use exhaustive search for small numbers of nodes and adapters, but the relatively low cost of heuristic search in such cases might make the extra complexity of supporting both styles of search unnecessary.

Figures 10 and 11 show that the latency of heuristic planning is below 90 milliseconds for 6 nodes with 14 adapters, and below 160 milliseconds for 12 nodes with 9 adapters. Exhaustive search would require many hours or days in these cases. Heuristic search failed to find the optimal plan for a four-node connection in only one case of 1000 tests, and in only three of 1000 tests for a five node connection. In 1 percent of the cases for six -node connection that were tested, an optimal plan was not

found. Thus, in 99+ percent of realistic cases, the heuristic search plan ner found the optimal plan. The cases where it was not found by heuristic search occur when the majority of connection nodes are able to run a very limited numbers of adapters.

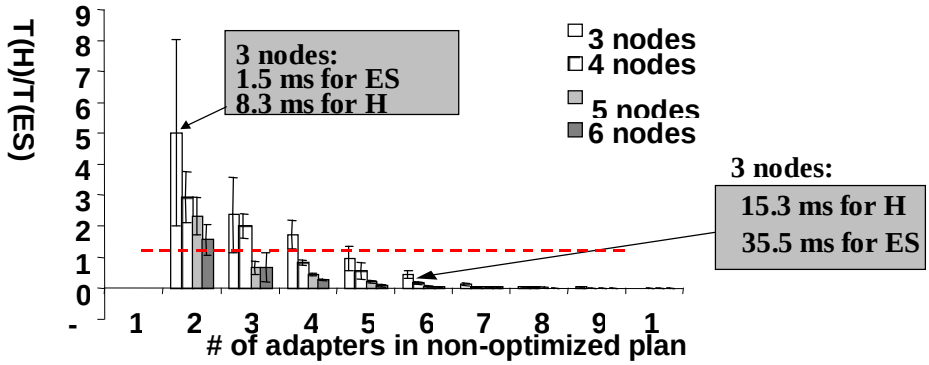

Fig. 9. Heuristic/exhaustive search latency ratio

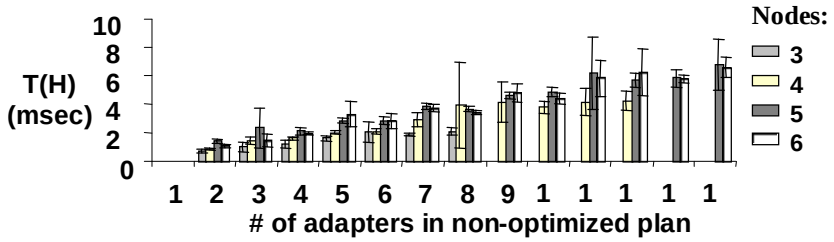

Fig. 10. Heuristic planning latency with respect to number of adapters and nodes

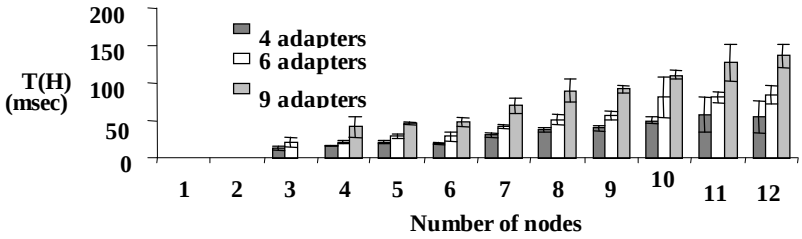

Fig. 11. Heuristic planning latency with respect to number of nodes and adapters

The Panda planning system was tested on true real-time applications. A video stream was generated by the WaveVideo multimedia package [3] and sent to a destination using a connection of HP 500 MHz machines running Panda. The efficiency of the transmission is measured in dB of PSNR (peak signal -to-noise ratio).

The graph in Figure 12 demonstrates the advantages of adapting video streams in low-bandwidth, often-insecure network links. The darkest bars represent a connection not using Panda. This connection's PSNR is 10dB lower than the Panda-adapted connections on low-bandwidth (150, 800 kbps) links. Even when Panda adds other benefits (encryption), the PSNR is still significantly better than not using Panda.

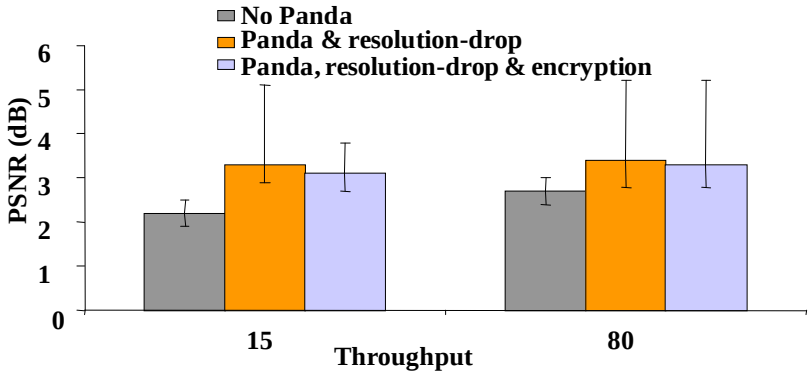

Fig. 12. PSNR (luminance) of adapted and unadapted (no Panda) video streams.

Figure 13 shows the advantages of centralized planning. One link has insufficient bandwidth, another is insecure. If the first link needs encryption and the second needs filtering, then the incremental plan puts an encryptor on the source node and a decryptor and a filter on the next node. This is worse than the optimized plan that puts the filter and encryptor on the source node and the decryptor on the next node. In the latter case, encryption and decryption are applied to fewer packets. This difference is measurable (Figure 14). Centralized planning displays better PSNR than local planning, and adaptation shows better PSNR than no adaptation.

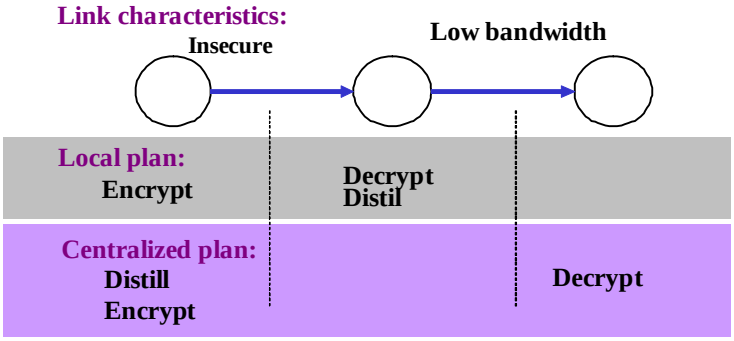

Fig. 13. Centralized versus local planning

6 Related Work

Many ONA technologies require applications to do their own planning, including ANTS [19] and SwitchWare [8], and the Rover tool kit [10]. Providing benefits to ONA-unaware programs usually require explicit user or system administrator configuration. In the Berkley proxy system [4], Protocol Boosters [13], and Odyssey [14], the services are typically simple and must be prelocated.

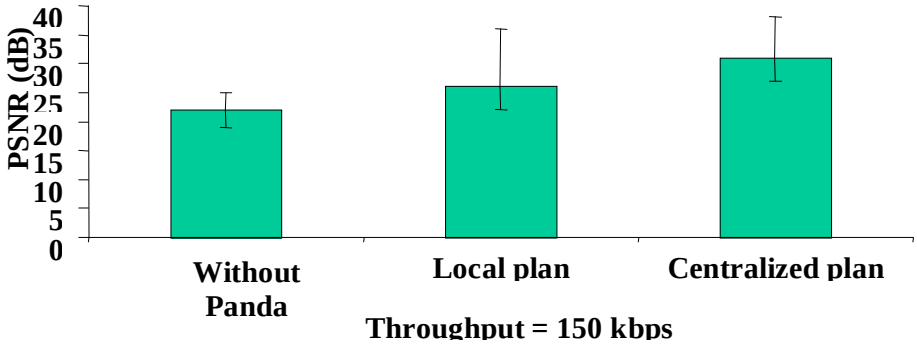

Fig. 14. PSNR of non-adapted, local planning, and centralized planning video streams.

Choi [2] uses automated planning route through the network. Their planning problem has the same complexity as the graph shortest-path problem. MediaNet [9] uses stream characteristics, user preferences, and templates to build a simple plan based on resource scheduling to improve performance and utilization. Conductor [20] is able to plug in a variety of plan formulation algorithms, but so far uses a relatively cheap and simple planning algorithm. The Panda planning approach is consistent with Conductor's planning model. CANS [6] plans based on high-level specifications of component behavior and network routing characteristics. The CANS algorithm complexity is $O(p^3n^3)$, while our algorithm has a lower complexity of $O(pn^2)$.

Planning is well studied in artificial intelligence and operational research [17]. Approaches include partial order planning in a solution space [18], recursive best-first search [12], and genetic algorithms [11].

7 Conclusions

This paper has demonstrated that it is possible to build a quick, effective, extensible automated planning system to bring the benefits of active networks to applications not written to use them. This system out -performs alternatives such as exhaustive search or treating the problems of each link separately. The system can be thousands of times faster than exhaustive search, and can handle problems too large for exhaustive search. Our system produced plans up to 100% better than incremental planning. The system can significantly improve the observable performance of real-world applications. Our Panda-based automated planner provided more than 10 dB PSNR improvement over

an unadapted data stream. These measurements also show the value of a sophisticated planning algorithm. The central plan produced by our complete algorithm provided around a 7 dB improvement over unsophisticated per-link planning. This planner could be used in a wide variety of open network architecture systems.

This planner is of particular value for peer systems. Such systems are likely to face multiple network problems on several different links, since each peer might be using a wireless network, a modem, or other problematic link. Peer systems often belong to unsophisticated users, and are unlikely to have good means of handling dynamic problems. Automated planning of adaptations avoids many such shortcomings.

Automated planning might assist design of distributed component systems, especially when they face unpredictable and dynamic conditions. For example, automated planning may support roaming services with drastically changing conditions. Our experience suggests that leveraging specifics of the problem space is vital to achieve success. While the planner outlined here is directly usable, the ideas and techniques used to create this planner may prove more even widely applicable.

References

1. M. Ancona, G. Dodero, C. Fierro, V. Gianuzzi, V. Tine, A. Traverso, "Mobile Computing for Real Time Support in Archaeological Excavations," *Proceedings of Computer Applications in Archaeology*, University of Birmingham, UK, April 1997.
2. Sumi Choi, J. Turner, and T. Wolf, "Configuring sessions in programmable networks," *Proceedings IEEE INFOCOM 2001*, vol. 1, pp. 60-67, April 2001,.
3. G. Fankhauser at al., "WaveVideo — An Integrated Approach to Adaptive Wireless Video," *Mobile Networks and Applications*, 4(4): 255-271, December 1999.
4. A. Fox, S. Griddle, E.. Brewer, and E. Amir, "Adapting to Network and Client Variations Using Infrastructural Proxies: Lessons and Perspectives," *IEEE Personal Communications*, September 1998, 5(4), pp. 10-19.
5. V. Ferreria, A. Rudenko, K. Eustice, R. Guy, V. Ramakrishna, and P. Reiher, "Panda: Middleware to Provide the Benefits of Active Networks to Legacy Applications," *DANCE 02*, January 2002.
6. X. Fu, W. Shi, A. Akkerman, and V. Karancheti, "CANS: Composable, Adaptive Network Services Infrastructure", USITS, March 2001.
7. John S. Gero and Vladimir A. Kazakov, "Evolving Design Genes in Space Layout Planning Problems," *Artificial Intelligence in Engineering*, vol. 12, 1998, pp. 163-176.
8. Michael Hicks and A. D. Keromytis, "A Secure Plan. Active Networks," *First International Working Conference*, *IWONA'99 Proceedings*, Springer-Verlag, 1999. pp.307-314.
9. Mickael Hicks, Adithya Nagarajan, and Robert van Renesse, "User-specified Adaptive Scheduling in a Streaming Media Network," IEEE OPENARCH'03, April 2003.
10. A. Joseph, A. deLespinasse, J. Tauber, D.. Giffort, and M. Frans Kaashoek, "Rover: A Toolkit for Mobile Information Access ," *Mobicom '96,* November 1996.
11. F. B. Kelly, "Routing in circuit-switched networks: optimization, shadow prices and decentralization ," *Advances in Applied Probability,* vol. 20, 1988, 112-144.
12. R. Korf, "Linear-Space Best-First Search," Artificial Intelligence, 62, 1993, pp. 41-78.
13. A. Mallet, J. Chung, and J. Smith, "Operating System Support for Protocol Boosters," HIPPARCH Workshop, June 1997.
14. B. Noble, D. Narayan, J. Tilton, J. Flinn, K. Walker, "Agile Application-Aware Adaptation for Mobility," *Proceedings of the 16th ACM Symposium on Operating Principles*, 1997.
15. L. Rizzo "Effective erasure codes for reliable computer communication protocols", Computer Communication Review, 27 (2), April 1997, p 24-36

16. sRudenko Alexey and Peter Reiher, "Experience with automated planning for Panda," Tech Report CSD-TR 010041, Computer Science Department, UCLA, 2001.
17. S. Russel and P. Norvig, *Artificial Intelligence*, Prentice Hall, 1995.
18. D. Weld, "An Introduction to Least Commitment Planning,"*AI Magazine*,.15(4), 1994.
19. D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: a toolkit for building and dynamically deploying network protocols," *IEEE Open Architectures and Network Programming*, Apr. 1998.
20. Mark Yarvis, Peter Reiher, and Gerald J. Popek, "Conductor: A Framework for Distributed Adaptation," *HotOS VII*, Rio Rico, AZ, March 1999.