Automated Planning for Active Networks

A dissertation prospectus

Alexey Rudenko

April, 2000

Abstract

Active Networks (ANs) are a relatively new technology for computer systems that allows dynamic deployment of services. The Internet is an obvious area that would benefit from fast deployment of protocols that would appropriately modify or reroute user data streams. ANs are meant to use fast-growing computational resources of modern computer systems as a relief for the resources of network communication channels with their often-limited capacity. The balance between communication channel and execution resources depends on user application requirements and network conditions at the moment of the communications. Complex network conditions in the conjunction with temporal constraints make automatic definition of necessary measures to improve the communications highly desirable. Automatic planning of AN services should be an important function of ANs.

We propose here the approach to the design of the planner for ANs. The approach is focused on the overwhelming number of problems of adaptation planning such as feasibility and efficiency of a plan, extensibility and composability of adaptations, and temporal limits. The planner automatically calculates properly ordered sequences of adaptations that modify user data. The purpose of these modifications is to increase throughput, reliability, and safety of communication channels.

1. Introduction

Traditional data networks passively transport bits from one end system to another. The network is insensitive to the bits it carries and they are transferred between end systems without modification. The role of computation within such a network is extremely limited, just header processing in packet-switched networks and signaling in connection-oriented networks. However, more extensive use of computations, provided by agents located in the network, can bring extra advantages to communications. The calculations must use computational resources efficiently and treat transferred data

properly. The amount of resources necessary for connection can be expressed with function *F(links)*, where *links* is a number of links:

$F(links) = \alpha \sum_{i}^{links} Link_i + \beta \sum_{j}^{links+1} Node_j$, where $Link_i$ is link *i* resources, for example throughput, latency, reliability, security, etc.; $Nodes_j$ is adaptation execution resources of node *j*, for example CPU cycles, memory, HD, etc.; and α and β are some coefficients. The goal is to redistribute link and node resources so that the following conditions were satisfied:

$Link_i < Link_{threshold,i}$, $i=1,links;$

$Node_j < Node_{threshold,j,}$ $j=1, links+1;$

$ResponseTime < ResponseTime_{threshold}$, where *ResponseTime* is a time of data transfer, *QoS* is a required quality of service, and *threshold* is a threshold value for each kind of resource that is predefined by user application and network.

The choice of the computations should take into account both user application and network requirements to compose a properly ordered set of adaptations for rerouting or modification of a user data stream. AN technology allows networks to dynamically deploy adaptations. ANs are samples of Open Architectures (OA), a broader field that allows improvement of the service through local adjustment of adaptations. Below we show a number of cases when the network would benefit from this technology.

Network technology and applications are changing rapidly, and existing protocols may not operate well in new circumstances. There are a number of examples (transit from IPv4 to IPv6, the rise of real-time and multicast communications using UDP, the rise of applications that frequently open and close TCP connections) that suffer from the inability of traditional networks to deploy and proliferate new protocols rapidly. AN technology would allow fast and relatively painlessly addition of new protocols.

Each link in a network may present a different level of bandwidth, latency, jitter, reliability, and security. This level depends on the current communication conditions of the network. The data transmission would be improved if it would be possible to adapt the data stream of a particular application to the network conditions. For example, data can be compressed to meet the requirements of reduced throughput of a poor-quality link, or data can be encrypted if security of a link is compromised. ANs should compose data-

modifying protocols that adjust the data stream to network conditions, keeping its integrity and security on the appropriate level.

Modern networks are characterized by the high level of heterogeneity. Conventional wired networks coexist with wireless networks, satellite networks, etc. In the future the level of network heterogeneity might even increase. A connection that traverses heterogeneous networks implies the coexistence of the number of communication protocols within the same connection. ANs allow composing a highly customized protocol that would serve the current connection best.

Some user applications have extra requirements for data streams. For example, palmtops have limited ability to receive and process images of certain formats. These applications may require special treatment of data stream on intermediate nodes, i.e. conversion the data from the original format to the one that is the appropriate for the palmtop. ANs allow the automatic deployment of special format-converting protocols serving the particular needs of that palmtop.

Automatic deployment of data-modifying protocols is the only way to serve the needs of legacy applications, i.e. applications that were designed without awareness of AN services. Legacy applications, which are still important for the existing market, cannot not use the benefits of ANs without special support. They are unable to recognize local communication conditions and therefore cannot keep their services on the appropriate level. Nor can they instruct the ANs on how to adjust to conditions. AN technology can automatically adjust the circumstances around these application to keep the performance above the certain threshold or degrade it gracefully. To achieve this purpose ANs should recognize their needs, decide how to satisfy them, and deploy the correspondent adaptations. Therefore, ANs should have access to the information about network conditions, which legacy applications usually do not have. The result of planning by ANs is a sequence of adaptations that should be applied to a data stream. The sequence of adaptations must preserve the integrity and security of data. That is achieved by the proper selection of adaptations and their ordering.

Planning is important tool of ANs that guarantees the consistency of adaptations and augments their efficiency. Efficiency becomes an issue because the execution of adaptations requires computing resources of intermediate nodes, which can be limited,

and increases the latency of data delivery. Unnecessary repetition of a particular adaptation as well as improper location of adaptations is highly undesirable. For example, assume that we have a connection that consists of more than 2 links. Assume that two links of the connection are poor and require data compression. It would be a poor idea to compress the data twice, once on each of the poor links. The data should be compressed once, before the first poor link and decompressed once, after the second poor link. This will save execution resources of intermediate nodes and save the time on one compression. Automated planning is intended to solve this problem.

The rest of document is organized as following: Section 2 contains related work. Section 3 contains the description of problems of automated planning. Section 4 contains the description of our approach to the design of automated planning including measurements and benchmarks. Section 5 contains the schedule of our future work on the implementation of the planner. Section 6 concludes the document.

2. Related work

In this section we will present a number of papers that indicate the current state of the research on Open Architectures (OA), ANs, and planning.

Active networks [Tennenhouse96] allow their users to inject customized programs into the nodes of the network. [Wetherall99] presents ANTS, a Java-based toolkit for constructing active networks. The transfer of adaptation code and the transfer of data are coupled in ANTS as an in-band function. ANTS limits the distribution of code to where it is needed, while adapting to node and connectivity failures. It improves startup performance and facilitates short-lived protocols by overlapping code distribution with execution. It allows customized processing to be expressed at a better granularity. The adaptation code will wait in a node cache for all subsequent data packets. ANTS presumes that applications are written specifically to use it. Legacy application packets will be treated as normal IP packets without active network service.

[Hicks99] is another example of an active network, SwitchWare. The SwitchWare active network architecture uses three layers: active packets, which contain mobile programs that replace traditional packets; active extensions, which provide services on network elements, and which can be dynamically loaded; and a secure active

network active router infrastructure, which forms a high integrity base upon which the security of the other layers depends. This security depends on integrity checking, cryptography, and verification techniques from programming languages. The authors of the architecture also designed a special language named PLAN for protocol design. Again, the basic system assumes applications explicitly invoke its services.

[Merigu99] presents an active network comprised of the CANEs execution environment and Bowman NodeOS. Bowman is constructed by layering active network services on an existing operating system. The host operating system provides low level mechanisms; Bowman provides a channel communication abstraction, an a-flow computation abstraction and a state-store memory abstraction, along with an extension mechanism to enrich the functionality. The CANE execution environment provides a composition framework for active services based on customizing a generic underlying program by injecting code to run in specific points called *slots*. Again, applications must explicitly invoke CANEs services.

[Noble97] presents Odyssey, application-aware adaptation as a collaborative partnership between operating system and applications. Odessey incorporates type-awareness for a data stream via specialized code components called wardens. To fully support a new data type, an appropriate warden has to be written and incorporated into Odyssey at each client. The wardens are subordinate to a type-independent component called the viceroy, which is responsible for a centralized resource management.

[Joseph96] presents the Rover toolkit, which combines relocatable dynamic objects and queued remote procedure calls to provide services for mobile applications. A relocatable dynamic object is an object with a well-defined interface that can be dynamically loaded into a client computer from a server computer to reduce client/server communication requirements. Queued remote procedure call is a communication system that permits applications to continue to make non-blocking remote procedure call requests even when a host is disconnected, with requests and responses being exchanged upon network reconnection.

The execution environments of ANTS, SwitchWare, and CANEs, the viceroy of Odyssey, and the applications that are designed using Rover kit do not contain a planner

as a tool that selects and orders their services as an integral part. Users must perform their own planning, typically at application design time.

Other open architecture systems seek to provide their benefits to programs and data streams that are unaware of the new possibilities. These application-unaware systems sometimes require explicit user or system administrator configuration, such as designating a proxy point, or pre-deploying various forms of adaptation modules. However, this approach limits their utility, since they provide benefit only when some person is intelligent and knowledgeable enough to foresee possible benefits and take appropriate action. Another approach is to automatically apply adaptations to data streams without explicit user intervention. At a limited level, this approach is already taken by protocols such as TCP, which does not demand that human users or applications assist it in adjusting to congestion on the line.

Protocol boosters [Mallet97] are software or hardware modules that transparently improve protocol performance. The booster can reside anywhere in the network or end systems, and may operate independently, or in cooperation with other protocol boosters. Implementation of boosters requires dynamic insertion of protocol elements into a protocol graph. In practice, protocol graphs are implemented as executable modules that cooperate via messages or shared state. Booster support requires inserting and removing the booster's function from the execution path followed for a group of packets handled by the protocol. As applications do not need to invoke protocol boosters explicitly, applications can be unaware of system services.

The Berkeley proxy system [Fox97] offers on-demand distillation that both increases quality of service for a client and reduces end-to-end latency perceived by the client. The system consists of three main components. First, the proxy is a controller process located logically between the client and the server. In a heterogeneous network environment, the proxy should be placed near the boundary between strong and weak connectivity, e.g., at the base station of the wireless mobile network. The proxy's role is to retrieve content from Internet servers on the client's behalf, determine the high-level types of the various components (e.g., images, text runs), and determine which distillation engines must be employed. Second, datatype-specific distillers are long-lived processes that are controlled by proxies and perform distillation and refinement on behalf

of one or more clients. Third, the network connection monitor, which determines and handles the characteristics of the client's network connection. Three methods can be used for this purpose: user preferences, network profile, and automatically-tracked values of effective bandwidth, roundtrip latency, and probability of packet error. The ;latter method, as the most complicated, was not implemented yet. Applications do not need invoke proxy services to benefit from them.

[Liuljeberg96] presents a set of enhanced services supporting the WWW, implemented as the Mowgli Agent, Mowgli Proxy, and Mowgli Data Channel Service. The most important features of Mowgli include more efficient protocols over the wireless medium, intelligent reduction of transmitted data, background transfers reducing the burstiness of traffic, and disconnected-mode support in the form of versatile user control over caching and cellular call setup. Mowgli provides three primary ways to reduce the transfer volume over the wireless link: data compression, caching, and intelligent filtering. Mowgli serves only WWW connections, which reduces the variety of services that are necessary to support the communication. A relatively small set of pre-computed plans will easily cover all necessary cases.

Conductor [Yarvis99A] and [Yarvis99B] demonstrates an approach toward selecting an appropriate set of adaptive agents and a plan for their deployment. Conductor allows arbitrary adaptations to be performed along the data path without reducing the reliability of the overall system. It includes a framework and a set of protocols for deploying adapter modules into a network. Conductor is fully transparent to applications, allowing easy addition of new applications and new network technologies. Conductor employs a unique reliability mechanism that allows a data steam to be arbitrarily adapted at multiple points, without compromising reliability [Yarvis00]. Conductor provides a centralized planning procedure that is run at the destination point of the connection using planning information on a fixed set of parameters for each link and node, user requirements specified in terms of link parameters and data characteristics, and the meta-descriptor and location of all available adapter modules. Although Conductor is able to plug in a variety of plan formulation algorithms, it currently employs a relatively cheap and simple planning algorithm that covers just simple cases, mostly because of an insufficient supply of well-developed planners.

Planner design for OA remains a barely explored area. Meanwhile, planning is a well known area in Artificial Intelligence and Operational Research.

[Dean94], [Nilsson98], and [Russel95] discuss different search strategies. The simplest way to build a planner is to cast the planning problem as search through the space of world states. Each node on the graph of possible states denotes some state of the world, and arcs connect worlds that can be reached by executing a single action. Partial order planning is the improvement of this search approach. In partial order planning, in the graph that describes the plan space, nodes represent partially ordered plans and edges denote plan refinement operations. Partial order planning as a refinement search within a solution plan space is presented in [Weld94], [Kamphampati94A,B], and [Ihrig96].

While classical planning has driven the majority of research in planning, more recently considerable attention has also been paid to planning in environments that are stochastic, dynamic and partially observable. To handle partially observable environments, information gathering is made part of the planning activity, and the classical planning techniques are extended to allow interleaving of planning and scheduling. Similarly, stochastic environments are modeled through Markov Decision Processes (MDP), and planning in such environments involves of constructing policies for the corresponding MDPs. [Kaebling95] and [Hauskrecht00] present techniques from operations research to bear on the problem of choosing optimal actions in partially observable stochastic domains. They introduce the theory of Markov decision processes (MDP) and partially observable MDPs (POMDP).

[Ling97] and [Bretthauer95] present a planning approach based on Constrained Resource Planning (CRP), which is a powerful tool for solving planning and scheduling problems using resource management focus. For example, in [Bretthauer95] a manufacturing system is modeled as an open network of queues and an optimization framework for capacity planning over a multi-period planning horizon is presented. The decision variables are the service rates (capacity) at each workstation in each time period. Capacity can be controlled at a work stations via the number of machines, modernizing or updating equipment, additional maintenance, the number of workers, the number of shifts, the use of the overtime, etc. The model involves the minimization of capacity expansion costs or the sum of product lead times to budget constraints on capacity costs.

[Gero98], [Jo98], and [East99] describe the application of genetic engineering based extensions to genetic algorithms for the layout planning problem. Genetic algorithms (GAs) are search methods inspired by natural genetics. The basic idea is founded on natural adaptive systems, where organisms evolve through generations to adapt themselves to given environment. Recent work on genetic algorithms has demonstrated their success in solving optimization problems, showing their simple but powerful search capability. Based on the advantage of GAs, genetic evolutionary concepts have been applied to the space layout planning and have shown promising results. GA approach was further developed in [Zhou97]. Evolutionary Computation (EC), developed on the base of GA, adopts natural coding such as float point or permutation naturally to represent the real- world problems and evolves them towards the optimal solution combined with the genetic operations. This approach was widely and successfully applied in a variety of research areas.

A somewhat different vision of the problem is presented in [Kelly88]. This paper considers the question how calls should be routed or capacity allocated in a circuit-switched network so as to optimize the performance of the network, using a simplified analytical model of a circuit-switched network. It is shown that there exist implicit shadow prices associated with each route and with each link of the network, and that the equations defining these prices have a local or decentralized character. The paper illustrates how these results can be used as the basis for a decentralized adaptive routing scheme, responsive to changes in the demands placed on the network.

We chose a partially ordered plan approach for our research, because it can be easily implemented and better fits the constraints and limits of planning for Active Network connections.

3. Problems in Planning

Planning consists of an action selection phase where actions are selected and ordered to reach the desired goals and a resource allocation phase where enough resources are assigned to ensure the successful execution of the chosen actions. An OA plan is a set of instructions to the nodes participating in a connection of what adaptation to use and in which order with respect to available node resources. The calculation of a

plan depends on many factors and is computationally complex. We will consider the most critical problems below.

3.1 Temporal factor

Real-time applications require very fast connection establishment, so the planning procedure must run with very strict temporal constraints. The plan must be created relatively quickly, since the data stream cannot be delayed indefinitely in search of the perfect plan. Depending on the specifics of the data stream, between microseconds and very small numbers of seconds are available to plan the remedial strategy. If the code implementing remedial actions is not ubiquitous, deployment costs must also be considered in planning.

The shorter connection life is, the faster the planning process must be. The temporal limits put additional constraint on existing planning approaches to satisfy on-line planning. A plan space of possible solutions for a particular connection can be very big, but the duration of the search within that space should fit the boundaries of communication-establishment time limits. The search should rely on some heuristics that are able to optimize for the most common cases. Using these heuristics simplifies the search, but at the same time it can cause the loss of good plans, which leads to poorer plans than with an exhaustive search. On-line heuristic planning presumes a possibility that a solution might not be found, in which case the connection cannot satisfy the requirements of quality of service or fails. This probability of a failure must be below the threshold of acceptable risk.

This temporal constraint plays the role of a global limit to the plan calculation process. If no feasible plan was calculated within temporal limit it means that planning process failed, otherwise the best found feasible plan must be returned even if much better plans might not have been evaluated yet. In case of a very strict temporal constraint the whole planning process can be stopped and an incompletely calculated plan returned as the only possible global plan. For a sufficiently long communication session, a cheap and inefficient preliminary plan can be calculated and deployed and the data transfer started. The search for a better global plan can continue in background, and an optimized global plan can be deployed later.

3.2 The consistency of adaptations

The system must sufficiently understand the format of the data stream that it intends to improve to take proper actions. In some cases, not only the format of the data stream must be considered, but also application end-points, hardware devices at those endpoints, and even wishes and needs of users. Certain adaptations are format aware, e.g. distillers, adaptations that modify the actual content of user data. For example, a colored image in format IMG must be transferred from node A to node B through an extremely poor link. Unfortunately, our distiller can understand only JPEG format. Then the planner should apply IMG-JPEG converter first, then apply the distiller, and convert the data to the original format using JPEG-IMG converter. The planner should be able to make all the analysis of data format consistency and format conversion whenever necessary.

3.3 The ordering of adapters

The system must be able to apply multiple remedial actions to the same data stream. The stream may encounter multiple problems at various points along the transmission path, and generally different actions will be required to solve those problems. Applying multiple actions implies that the system must be able to determine if a set of actions are compatible. The canonical example of incompatibility is to meet problems of security and inadequate bandwidth by first encrypting the data stream, then ineffectually compressing the encrypted version of the stream. The difference between this case and the case of the data format consistency is in this latter case there is no conflict of data formats. Compression can be correctly applied to encrypted data, it simply fails to achieve its goals. The only problem is ineffectiveness in using the adaptations in one order and effectiveness in using them in opposite order. It presumes a certain extension of the notion “format” that expresses the “compressibility” of data. As compressibility of data decreases significantly after encryption, which makes the later compression useless, the planner should make the decision to put the compressing adaptation before encryption. The compressing adaptation also reduces the compressibility of data, but the encryptor is indifferent to this aspect of data format, unlike the compressing adaptation, and will perform efficiently. The development of these aspects of data format is very important for planning. As these parameters may

change and increase in quantity frequently, the planner should be independent of them as much as possible.

3.4 The efficiency of the plan

It is important to save execution power of network nodes and links. Adaptations should be applied in the most efficient way, i.e. without repetitions, minimizing the use of node and link resources. For example, we have a connection that consists of three nodes A, B, and C, connected with two links AB and BC. Assume that link AB requires two-fold compression of data, and link BC requires 4-fold compression of data (see Fig. 1). The optimal plan should apply compression only once, at node A with 4-fold compression, which will satisfy the requirements for all links.

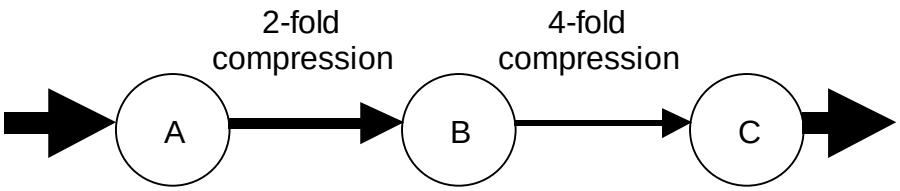

Fig. 1: 2-fold and 4-fold data compression

Another way to increase the efficiency of adaptations would be extending the effects of an adaptation that reduces the amount of resources needed for the connection. For example, a peer-to-peer connection consists of a number of nodes and links that connect them. One of the intermediate links requires compression of the data. The whole network will benefit if the compression will be run on the source and decompression on the destination of this connection, because of the totally reduced amount of data transferred. The problem is that the system must be able to determine if the open architecture is willing and able to run all adaptations that the system proposes at the locations it chooses. Some adaptations might not run properly on particular nodes. Some nodes might be unwilling to run particular kinds of adaptations. Perhaps some

nodes limit the resources expendable on a single packet or entire data flow at that node. These constraints may cause a different set of adaptations to be performed, or may affect where adaptations are located.

Optimization of a plan presumes an extensive search in the space of possible plans. The plan that demonstrates the best use of network communication and executional resources must also be deployable. A plan that is deployable is called *feasible*. As we have shown in the previous example, the best location of a compression adaptation would be the end points of the connection. But the plan might not be feasible if the endpoints of the connection are unable to deploy the adaptation because of insufficient resources, lack of access to the chosen adapters, etc. The costs of the deployment of a plan can be another factor of optimization.

The planner also should take into account the fact each adaptation applied to a data stream adds delay to that stream. Even if every adaptation alone does not trespass the threshold of latency of data transfer, all together they can make the latency unacceptably long. The planner should try further compression of the transferred data, choose less time-consumptive adaptations, run adaptations on more powerful servers, etc.

Optimization of the plan is an important function of the planner, which also should be able to find a feasible plan within the limited temporal interval.

3.5 The extensibility of the system

Extensibility of planning is a very serious problem for the planning system. New transformations, data formats, and constraints that are unknown today can appear in the future. The need to handle both existing network problems and problems that will be discovered in the future will produce a significant number of different adaptations, written by many parties. The design of the planner and adaptations should presume some common interface, well understood by both sides. Creating or using a plan requires knowledge of the available adapters. The planner should know their names, locations, and how to use them. The planner should distinguish the versions and specifics of adapters. The planner should know the amount of resources that those adapters require.

If the compatibility of the planner and adaptations that exist already or will be created in future is lost, the whole idea of planning in OA will be seriously shaken. The only way of planning would be to accompany each planner with its own libraries of

adapters. It would make life of a planner designer a lot easier, but we believe that in practice the adapters and planner will be maintained independently.

3.6 Other problems

The separate maintenance of the planner and adapters immediately raises more problems of adaptation composability in addition to the described in previous sections [Zegura98].

Errors can occur during the process of planning, deployment or running the adaptations. The planner should be able do error handling to preserve the safety of data and re-run the planning process to create error-free plans.

The composition of adapters raises also security issues. Some adapters will require that the user be authorized to execute the adapter. In some cases the process of planning will require that the designer of the adapter be trusted. The discovery of the resources necessary for the deployment of the plan may be the subject to security-based access constraints.

The use of some adapters will need to be monitored for accounting purposes.

We are not planning to handle these issues; they can be addressed to planner designers in future.

4. The approach to planning for OA

Our goal is to implement a planning procedure for peer-to-peer communications. We are working in the context of an application-unaware active network support system called Panda. However, the planner could be applied to other OA systems, e.g. Conductor. This work is focused on automated planning aspects of active network services. Below we give the description of the environment where the planner should be implemented and our approach to the solution of the planning problem for peer-to-peer connections.

4.1 The description of our environment

Active networks [Tennenhouse96] represent a new approach to network architecture that incorporates interposed computation. These networks are “active” in

two ways: routers and switches within the network can perform computations on user data flowing through them; furthermore, users can "program" the network, by supplying their own programs to perform these computations. Active networks allow users to deploy new services by tailoring components of the shared infrastructure to suit their requirements. Active technologies have been emerging in the fields of operating systems and programming languages for some years: PostScript, Safe-Tcl, x-kernel, etc.

The architecture of an active network node consists of three basic components: The Node OS that supports communication channels; the execution environment (EE) that exports an API or virtual machine that users can program and control and provides an interface through which end-to-end network services can be accessed; and the agent that performs user application message handling. The message created by the sending application goes to the Node OS, which determines which of several EEs should handle the message, or directly to EE as for example in ANTS. The EE in turn may choose to select an agent or other piece of code to handle the message. Once the agent completes the handling, the EE calls the Node OS to request actual physical transmission along some network link. Intermediate nodes can adapt the message via local agents. At the destination node, after the EE and agent have done their work, instead of requesting further transmission, the EE requests the delivery of the message to the destination object.

ANTS is an EE for an active node transfer system distributed as a Java-based toolkit for constructing an active network and its applications. We use ANTS as a base for our research. It provides a programming model that allows some kinds of protocol processing to be expressed, a code distribution system for loading new protocols into the network, and node runtime for executing them. ANTS is intended to support novel network services for routing, caching, transcoding, combining, filtering, regulating, and otherwise processing packets within network itself. This includes the notion of "application-specific" protocols where portions of the application processing are "pushed" into strategic nodes of the network. ANTS design is guided by three goals:

1) the nodes of the network should support a variety of different network protocols being used simultaneously,

2) new protocols should be supported by mutual agreement among interested parties, rather than requiring centralized registration of agreement between parties

3) new protocols should be deployed dynamically and without the need to take portions of the network “off-line”.

The combination of a packet and its forwarding routine is called a “capsule”; the forwarding routine is executed at every active node the capsule visits while in the network. The forwarding routine can be cached on active nodes and be executed for a particular data stream. The design of ANTS presumes that applications that use ANTS must be aware of it.

Panda [Reiher00] provides an application-unaware active network. Panda automatically traps non-active data streams and converts them into streams of active packets. Panda also creates plans for which active services should be performed at each active network node or switch along the path. A Panda prototype has been used in our lab for over a year. The planning capabilities of the original prototype were extremely primitive. We are currently implementing an improved prototype that will offer better planning support. Panda currently works with the ANTS EE.

Active services in Panda are implemented as adapters that should be deployed according to a plan on nodes designated by the plan. After the plan is activated the adapters modify all data packets arriving at the node. Adapters increase the cost of the connection, using resources such as CPU cycles, storage, network controls, etc. The deployment of adapters also requires extra time and network bandwidth. Two adapters may have different characteristics even if they do the same kind of adaptation for a data stream. For example, one adapter might compress a data stream by converting color images to black-and-white images; another adapter can achieve the same level of compression by reducing the resolution. The choice of a particular adapter in this case depends on the requirements of the user for the data stream. The location of an adapter affects the characteristics of data stream. It is more profitable to extend the number of links covered by some adapters. For example, assume that some link requires an adapter that uses the Ziv Lempel technique for data compression. Then locating compression and decompression components as far from each other as possible will improve overall communication because each link will benefit by forwarding compressed data, even if it has sufficient bandwidth. At the same time it is undesirable to extend the effects of a

forward error correction adapter on links that do not actually require extra reliability, because this adapter increases the amount of data that must be sent.

The Panda prototype consists of three basic components (see Fig. 2). The Panda Interception Component (PIC) traps messages sent by applications that do not use active networking capabilities. It examines such messages and gives those it thinks Panda can assist to the Panda Adaptation Component (PAC). The PAC is responsible for planning which adapters to use on behalf of a given data flow and deploying them at the proper locations in the network. The planning function of the PAC requires information about conditions in the network. The third Panda component, the Panda Observation Component (POC) provides this information. The POC observes network and node conditions and provides information to the PAC as required for planning. If conditions change drastically, the POC can signal the PAC, which may choose to abandon the existing plan and re-plan. Panda must be deployed at any node where adapters are to be run. Panda planning requires information and cooperation from all Panda nodes traversed by a data flow.

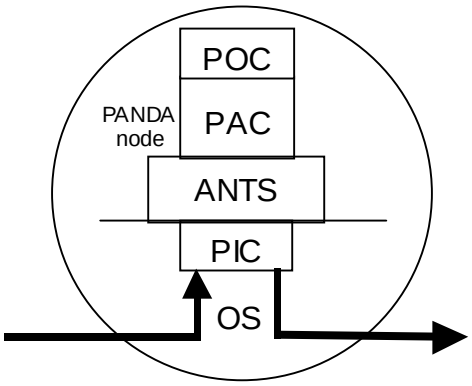

Fig. 2: A PANDA-enabled node

4.2 Planning

In Section 3 we defined a plan as the instruction to the nodes participating in a connection of which adaptation to use and in which order. The basic problem of planning is to find remedies to a given set of problems located at particular nodes or links in data path.

4.2.1 Basic approaches to planning

One simple solution is to precompute a set of reusable plans suitable for common circumstances. The planner need merely find a match for its observed problems from among the set of precomputed plans. This solution has the advantage of requiring a very unsophisticated planning component, but the disadvantage of little flexibility. It can only deal with specific sets of problems we foresee. The basic idea can be extended to allow precomputed plans with slots to be filled in by the planner at runtime [Merigu99], or by allowing the planner some flexibility in where to locate adapters. The more powerful the extensions, the greater the flexibility, but the greater the complexity of the planner.

Looked at another way, finding a good plan is a searching problem. Remedies solving particular problems must be found, and they must be located on nodes that are properly positioned in the data stream and that offer sufficient resources. There are a finite number of problems, a finite number of remedies, a finite number of potential remedy locations, and a finite number of constraints on what can be done at a given location for each flow. The combination of these factors defines the space of all possible plans for each flow that can be calculated. Feasible plans, the plans that can be deployed and actually fix all connection problems, are an interesting subset. Some feasible plans are closer to optimal than the others in terms of the efficiency of the data transfer and the resources needed to deploy and run the adaptations.

One can imagine functions that define the value of certain solutions, based on whether they solve the problems faced and the costs they incur in doing so. The evaluation function must calculate a certain numerical interpretation of all factors of network communication and adapter deployment, such as throughput. Monetary cost of the use of the links in the connection is another important factor in the evaluation function. The evaluation function must also take into account the execution resources of the nodes that run adaptations, as well as the cost of deploying adapter code. A search algorithm could evaluate various possibilities to find the optimal solution, or at least a feasible solution that solves all problems at an acceptable cost.

There is an obvious tension between providing optimal behavior for a single flow and providing overall optimal network behavior. We do not presume to offer fresh insight on this problem, but suggest it can be limited by the commonly chosen means in

active network research, limiting the resources devoted to a given flow network wide. Additional research in this area is ongoing in the active network community, and we will leverage this work.

We consider the number of nodes that run the adaptations as an important factor of the search strategy. The fewer the nodes running the adaptations, the smaller the solution space and the easier it is to find the optimal plan. However, the set of the plans that are built on a smaller number of nodes might not contain feasible plans due to a lack of resources. For instance, if only the endpoints are considered, a PDA at one endpoint may have insufficient memory or CPU cycles to adapt a video stream in real-time. Also, plans built on a limited number of nodes might not be feasible because they do not include a particular node required by an adapter. For example, the adapter that provides an digital signature must be applied exactly at the node whose digital signature is needed and cannot be replaced by somebody else's signature. At the same time, if all nodes are to be considered some other strategy for reducing the search space is needed.

Looking at the problem as an example of search suggests some solutions. The most obvious is an exhaustive search. If an evaluation function properly values the costs and benefits of applying various candidate solutions, exhaustive search will find the optimal solution. If the number of candidate solutions is small enough, exhaustive search is a fine method. Consider, however, a data stream that faces four problems, with 256 different adapters available. Assuming one adapter is required to solve each problem, and a purely exhaustive approach to deciding which adapters to use, the system must examine over 4 billion possibilities. If the problem of adapter ordering is considered, or the problem of where to locate adapters is added, the possibilities grow.

One quick way to limit the growth of the search space is to encode adapters with the problems they solve. Instead of blindly trying all possible adapters for all possible problems, the planner can consider only adapters known to solve the particular problems being faced. In the example above, if the 256 adapters each solve a different problem, the only issues an exhaustive planner would face would be ordering the adapters and locating them on particular nodes. For a small enough number of adaptation locations, the total number of possible solutions would be reasonable. But if there are 25 different data compressors, 12 encryptors, half a dozen error-correcting encoders, and three or four reservation schemes, the number of possible solutions skyrockets. Part of the promise of

open architectures is that they will allow a proliferation of adapters that help data streams, so designing a system suitable for only a small number of adapters seems short sighted.

Non-exhaustive search strategies can start from an initial candidate solution and attempt to move towards a better solution. One initial solution is to do nothing, with each search step being the addition or replacement or relocation of an already chosen adapter. A different initial solution is to deploy some remedial adapter in the immediate vicinity of each problem. Each search step would be to replace an adapter or move it to a different location. Another initial solution is to locate all required remedial adapters on the source and destination nodes, with each step relocating adapters to more appropriate locations. The amount of work done to find superior solutions must be limited by the amount of latency acceptable to the user. If the planning process takes too long, simply sending unaltered data over an unassisted path may be better. However, if one of the as-yet-undetected problems turns out to be insufficient security, this decision could be disastrous.

4.2.2 Local, centralized, and distributed planning

We distinguish three kinds of planning: centralized, local, and distributed. Centralized planning is based on the planning information for all links and nodes of the connection. It runs at one node and produces a complete global plan for the whole connection. Local planning runs on each node of the connection and creates a plan just for two neighboring nodes and the link between them. Distributed planning runs on a number of nodes that participate in the connection. After some period of negotiation the nodes produce the global plan for the connection.

As we saw, the search through the plan space can be a time-consuming operation that due to the time limits of the planning might be unsuccessful. This is the main disadvantage of centralized planning. However, the space of the search can be sufficiently reduced if a plan would be calculated locally, just for the link where the problem occurs. The consecutive calculation of local plans for each neighboring pair nodes of the connection will produce an *incremental* plan. Local planning is relatively fast as it consists of trivial plan-per-link plans. Incremental planning presumes that each node that participates the communication session builds its own local plan being aware of

up-stream planning data and the plans built by up-stream nodes. Nodes cannot change any decisions made upstream.

The main disadvantage of incremental planning is the inefficiency of the overall plan. For example, our communication consists of source node A, destination node C, and intermediate node B (see Fig. 3). Assume that the user application needs a throughput of 10Mbps. Link A-B has spare throughput of 5Mbps, and link A-B has throughput of 2.5Mbps. A must reduce the user data stream by half, from 10Mbps to 5Mbps. It builds the corresponding plan and deploys the adapter that compresses user data by half. B obtains planning data from up-stream. It knows that the user desires 10Mbps, but the A-B link can supply only half of the necessary throughput. B deploys the adapter that decompresses data back to 10Mbps. Then B deploys a compressor that reduces the data by three quarters. C deploys the adapter that will decompress data back to 10Mbps. As we see, the data was compressed and decompressed twice instead of using only one 4-fold compression on A and one 4-fold decompression on C. Incremental planning produced a highly inefficient planning solution. Only a global planning protocol, centralized or distributed, can notice this kind of inefficiency and instruct A and C to run the correct compression.

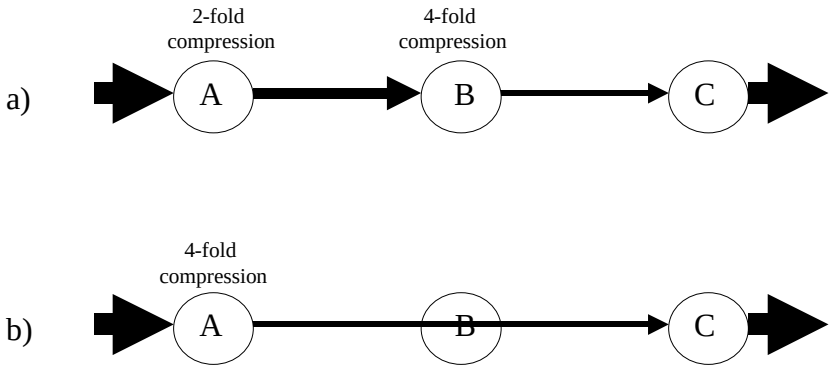

Fig. 3: a) inefficient plan b) efficient plan

Combining both incremental and centralized planning can give the advantages of both approaches. A communication can use an incremental plan first, which can be deployed relatively fast, and then switch to a centralized plan whenever it is calculated and deployed. The cost is that the system must support clean switching between plans.

Distributed planning can be implemented using sophisticated negotiating protocols between the planners that participate in the connection. Like most distributed solutions, it is more complex than a centralized solution. It goes beyond the scope of this work.

4.2.3 Planning information gathering and storing

We can presume that any active network node collects and stores planning data about its neighbors and adjacent links. Any local plan can be executed without a special planning information gathering procedure. The circumstances are different for centralized planning, which must collect planning information about all nodes and links that participate in the connection. The process of planning data gathering should occur on-line, during the handshaking phase, before the connection is established. The first node that obtains planning data for the whole peer-to-peer connection is the destination node. We believe that the centralized planning process should be run as soon as possible, and therefore the destination node is the most appropriate place for it. After the centralized plan is calculated it should be distributed among other active nodes that participate in the connection.

4.3 The steps of the planning

Based on our analysis, we will combine incremental and centralized planning. These are the steps of planning procedure:

1) The initial planning signal is sent from source to destination (see Fig. 4). It collects planning data while it is traveling. The data is delivered to the planner.

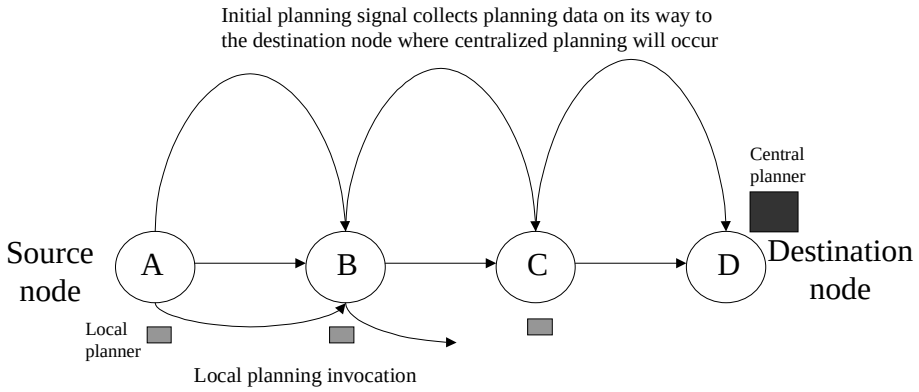

Fig. 4: The local and the centralized planning. The initial planning signal and local planning invocation start on the source node at the same time. The first one moves faster than the second one because it only collects the planning data. The plan invocation signal waits for the adapter selection by a local planner and then moves to the next node. The planner orders the selected adapters and deploys them as a local plan, as one step in the incremental plan. The central planner collects the planning data, calculates the central plan and sends it to the nodes for the deployment, replacing the incremental plan.

2) Parallel to the initial planning signal, local planning is invoked. It occurs incrementally, link by link. When adapter selection is finished on one node, the result of it is sent to the next node downstream. Then the node completes its local planning process i.e. orders the selected adaptations and deploys them.

3) When the next node obtains the list of provided adaptations from the previous node it can start its own local planning process. It is important for the next node to be aware of irreversible actions, such as lossy adaptations, that are planned by the previous node. It is undesirable to plan another color-to-B/W conversion to user data if it was applied already on one of the previous nodes.

4) When the local plan of the link adjacent to the destination is deployed, acknowledgement is sent to the previous node upstream. When the plan of the previous node is deployed and acknowledgment from the downstream node is obtained, the acknowledgement continues upstream. The process continues until the source obtains the acknowledgment from its downstream node that indicates that all local plans are deployed. Then the source starts the data transmission. Note that the chain of local plans can be considered as one global plan that is consistent but not optimal. If one local planning calculation fails because of insufficient resources on the node, the whole local

planning procedure fails. The only hope would be that the centralized planning would succeed.

5) At the same time data collection packet reaches the central planner, it starts the global plan calculation process. When the global plan is calculated it must be deployed.

The local plan calculation consists of two steps:

1) Adapter selection
2) Adapter ordering

The centralized plan calculation consists of three steps:

1) Adapter selection
2) Adapter ordering
3) Optimization of the global plan

The first two plan calculation steps, adapter selection and adapter ordering, are similar for both planning concepts. They produce a global plan as a chain of consistent local plans. These global plans obtained by local planners and a centralized planner might be different because local planners and the central planner may select different adapters to solve the same communication problems.

The adapter selection process is implemented through a database search among adapter packages. A special interface between the planner and adapter packages will be designed for this purpose.

Adapter ordering is a planning problem that can be solved through *least-commitment* planning. In least commitment planning, a plan is defined by a partially ordered set of adaptations referred to as a *partially ordered plan*. Instead of searching in the space of possible states, least commitment planning searches in the space of possible partially ordered plans. The order in partially ordered plans should come from heuristics, i.e. the experience in the ordering of adaptations. For example, experience shows that compression should be applied before encryption. Then in a plan we can add an order to a fully unordered set of adaptations: compression and all adaptations that run before compression should be applied before encryption and all adaptations that run after encryption. The complete order can be added to the plan from particular network conditions and user preferences.

Centralized planning before running the optimization phase builds an initial global plan using local planning. The chain of local plans obtained through the selection and ordering of adapters establishes a sub-optimal global plan. This plan can be optimized through the reduction of link and node resources that it takes to execute the plan. The optimization can be achieved via:

- Extending the effects of the adapters that reduce the usage of link resources, such as compression, distillation, filtering
- Merging adapters that duplicate their effect on user data, saving node computational resources. For example, compressions running on two adjacent links can be substituted with one compression adapter covering both links

The problem of plan optimization is very complicated because of complicated interactions between the effects of adapters on user data and the large complexity of the operation, which is exponential. The set of heuristics can sufficiently reduce the search tree, but we still cannot be sure that the most optimal solution can be found, especially if we take into account the time constraint, which is limited by seconds. That is why we rely on randomized heuristic search. The idea is to try randomly chosen branches of the search tree to a certain depth, compare the results using an optimization function that shows the amount of resources used, and continue the search on the branch that promises the best result. Obviously it cannot guarantee that the optimal result, the point where evaluation function has its global extreme, would be found. The techniques, such as simulated annealing, that help to find global minimum/maximum may not help as our optimization model is discrete and changing the size of the step may not be possible.

The technique that is applied to the optimization process is similar to refinement planning. The idea of the algorithm is the following. The chain of local plans is a partial plan. The process generates candidate global plans through merging neighboring plans. We start building the merged plan by choosing a local plan and trying to merge it with neighboring plans, one by one, using heuristics that are based on our experience and knowledge about the properties of adaptations. The intent is to extend adaptations that save link resources, not to extend the adaptations that use more link resources, and merge similar adaptations to save node computational resources.

The constraints on both plans to be merged must be preserved. A conflict between the constraints must be solved through adding new steps to the plan. If the conflict cannot be solved, the merge is canceled, and the optimization process is resumed at the local plan that refused to join the merged plan. Ambiguities require the consideration of all possible cases. When two plans are merged we evaluate the result applying an evaluation function. The newly obtained global plan should be closer to optimal than the original. The process continues adding neighboring plans one by one until, in the ideal case, all plans are merged. The process can be interrupted at any moment; the merged plan and the set of local plans that were not merged, given that available node resource constraints are satisfied, can be returned as the result of plan optimization procedure. Typically, the reason for interrupting the optimization process is the temporal constraint, i.e. the urgency to deploy the plan. However, the success of plan merging depends on correct information about adapters, particular network circumstances, the plan merging procedure, etc.

For example, assume that we have a network that consists of three links. The first and second links can use any encryption, but the third link requires special encryption that can be used only on that link. Incremental planning build local plans on the first and second links that run encryption on each. The local plan on the third link contains special encryption. After optimization the encryption on the first and second links was combined. The third link encryption was not combined with it because of its special requirements.

Because of resource constraints, the result of the merging of the local plans may be dependent on the selection of the initial local plan, and it is impossible to say in advance which initial local plan should be chosen to bring the optimal result. That is why we choose the initial local plan randomly, with some probability. The probability may not be distributed uniformly among the local plans; the choice of the initial local plan depends on the feasibility of the local plan, the ordering of adaptations that form the local plan, etc.

Figure 5 illustrates how the selection of the initial local plan influences the result. Assume that we have a connection that consists of three links and four nodes A, B, C, D, and E. End points of the connection A and E are able to run one adaptation each and the

intermediate nodes B, C and D are able to run two adaptations each. A problem of link AB requires adapter X, a problem of link BC requires adapter Y, and a problem of link DE requires adapter Z. The adapters X, Y, and Z save the resources of the connection, and it is desirable to extend them as much as possible. Adapters consist of two parts each: DO and UNDO, e.g. compression and decompression, encryption and decryption. Adapter X if applied on the same node with adapter Y should be executed first. It is easy to see that the selection of either of adapters X, Y, or Z as an initial leads to absolutely different plan.

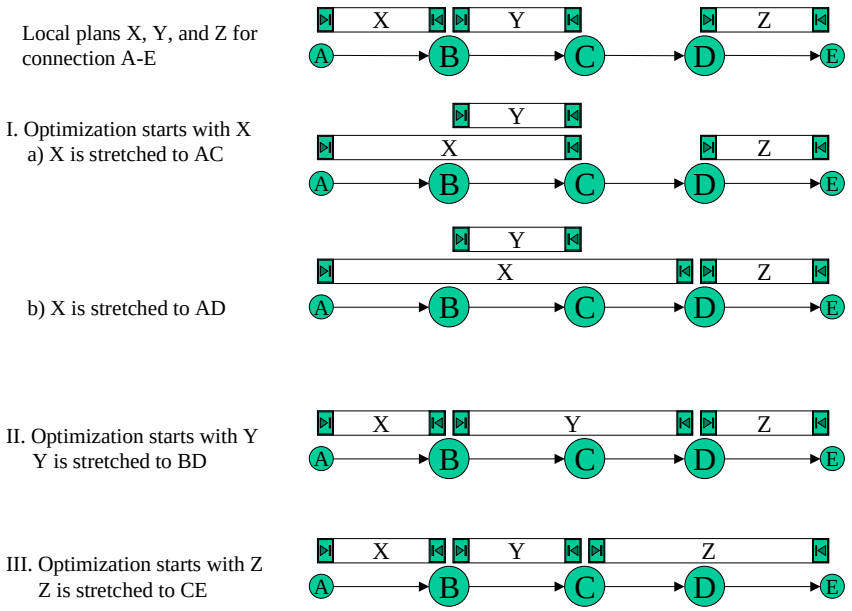

Fig. 5: Optimization that starts with different initial local plan after merging produces different resulting plans

As it was mentioned above, the number of merging processes starts with different randomly chosen local plans and continues up to a certain depth. The results of this preliminary search are compared by the value of the evaluation function, and then the process resumes the search on the most promising branch. The process is terminated when the search reaches a dead end, or planning time is over.

4.4 An example of planning process

Assume that we have a connection that consists of three links and 4 nodes A, B, C, and D. AB is a wireless link with limited throughput. It is also unreliable and

insecure. Link BC is a dial-up link. It has even lower throughput than AB. CD is a wired Internet link with sufficient throughput, but insecure.

When node A starts a connection with D, it sends initial message. The message moves to D collection planning data. Node A also activates local plan process for link AB. The planning process starts with the selection of adapters. The selection of adapters starts with search in planner database for a adaptation package that contains the correspondent adapter. When an adaptation packages is selected, the search in adaptation package database occurs. The planner selects actual adaptations that can resolve link problems. After the selection of the adaptations, node B starts the local planning process for link BC. At the same time planner orders the adaptations for the local plan AB. When the adaptations are ordered, the plan AB is ready to be deployed. Nodes A and B deploy the correspondent adaptations. In our case it is LZ compressor, FEC, and encryptor. Local planners for local plans BC and CD work in the same way as one worked for AB. Local plan for BC contains LZ compressor and Color Dropper; local plan for CD contains only encryptor. Every local planner order adaptations using pre-calculated templates for single link ordering. The ordered adapters for local plan AB are ZL compression, encryption, and FEC. The ordered adaptations for local plan BC are Color Dropper and ZL compressor. The local plan CD contains only one adapter. After the deployment incremental plan is ready for use.

At the same time when initial message with all planning data reaches node D, the centralized planning process starts at D. The process of adaptation selection and ordering for centralized planning works using same methods as incremental planning. Figure 6 illustrates the process of the centralized planning. The planner at node D selects and orders the adaptations for all four nodes, see Figure 6, b) and c). Initial global plan consists of local plans AB, BC, and CD; same initial global plan is on Fig. 6, c) and Fig. 7, c). Note that, unlike compressor, FEC, and encryptor that consist of “DO” and “UNDO” pieces Color Dropper is a lossy adapter that contains only DO part. On the Figure 7 it is shown as the short bar that does not cover the whole link. Optimization of the initial global plan is on the Figure 7, b) and c). The result of the optimization is a global optimal plan. Compressor and encryptor are to be executed on the end points of the connection because compression reduces overall throughput requirements for the

connection; encryptor augments the security of the whole connection. However, FEC runs on link AB only because it increases the requirements to channel throughput, and it is undesirable to extend it.

When the global plan is deployed, node A switches the data stream from incremental plan to the new optimal plan. When the connection is no longer in use the plan should be decommissioned.

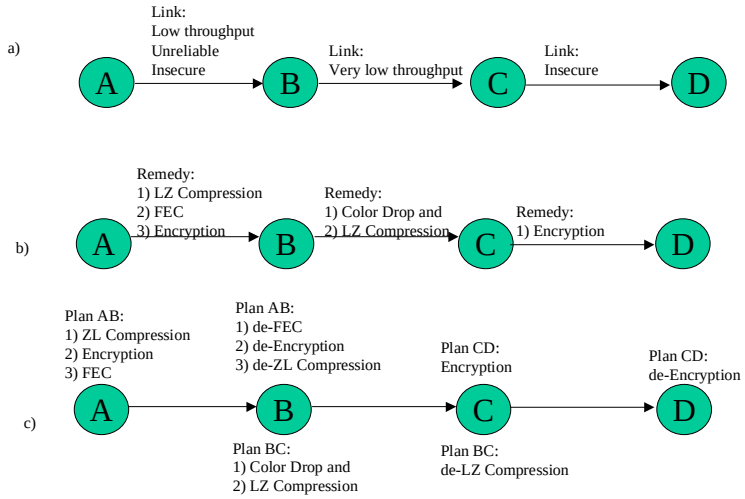

Fig. 6: a) three links, four nodes connection; b) selection of adaptations; c) ordering of adaptations

4.5 The implementation of the planner

We will implement the planner on the top of ANTS. Panda and the planner will work in parallel as two independent applications. The planner will calculate plans; Panda will deploy them using the ANTS EE. The source Panda node will query the local planner to obtain a plan for a connection that must be established. The planning information must be collected by Panda nodes and delivered to the planner. The planner should activate the planning process involving planners that belong to other nodes that participate in the connection. The calculated plan is delivered to the Panda nodes that must deploy it. During the connection more than one plan can be implemented and the data stream must be properly switched from the old to the new plan.

The planner invokes local and centralized planning. As centralized planning is time consuming, the data transfer will start with the local plans that should be deployed on Panda nodes as a chain. When the central plan is calculated and deployed, the source Panda node will switch the data stream to the central plan, and local plans should be decommissioned.

Local planning will consist of three consequent steps: adapter selection, adapter ordering, and plan feasibility verification. Centralized planning will consist of four consequent steps: calculation of per link plans that consists of the adapter selection and the adapter ordering, plan optimization through the merging of the link plans, and plan feasibility verification.

Plan optimization through the merging of the link plans intends to extend adaptations that save link resources, not to extend the adaptations that use more link resources, and merge similar adaptations to save node computational resources.

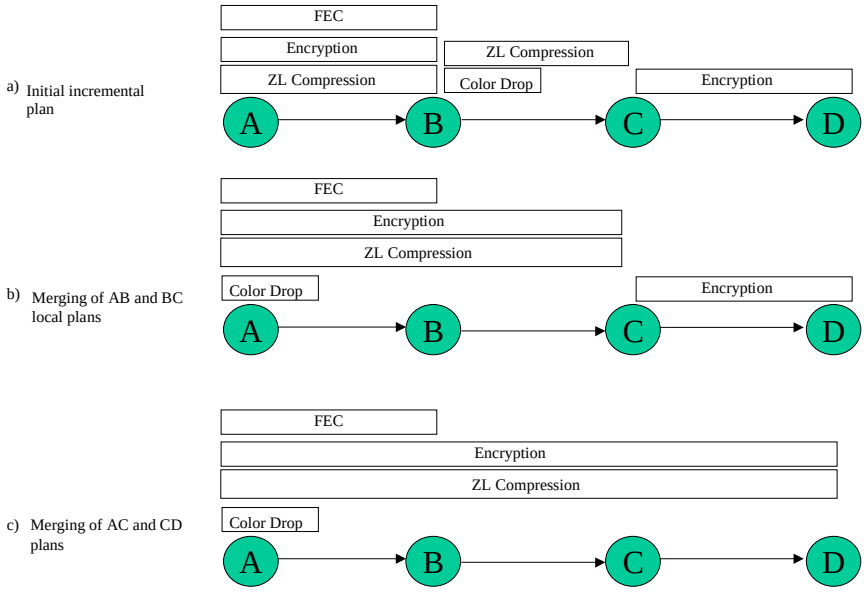

Fig. 7: Plan merging starting with the most left local plan

The proposed implementation of the planner is focused on the following issues:

1) Planning protocol

The planning protocol gathers and stores planning data and deploys a plan based on a Panda node. The protocol is based on the combination of incremental and centralized planning. The message that collects the planning data for the centralized

planning and invokes the centralized planning is followed by the incremental planning message that invokes the incremental planning. Plan deployment is supported by acknowledgments to planners and to a source node. If all nodes that participate in the connection report that their fraction of the plan is deployed, then the source node switches the data stream through the newly deployed plan.

If an acknowledgement to a planner and a source node indicates that a local plan cannot be deployed successfully on a particular node, the planner and all downstream planner should re-plan, and the source node should wait for the new acknowledgements from all downstream nodes. When all local plans are deployed the source node starts the data transfer.

If a node reports to the destination node and to the planner that it cannot deploy its fraction of the centralized plan, the planner re-plans and tries to deploy the new centralized plan. The source node waits for acknowledgments from all nodes of the connection, then switches the data transfer to the new plan.

2) Selection of adapters

The selection of adapters uses the planner-adapter package interface. A planner contains a database of all accessible adapter packages. From that database the adapter can find which adaptation package resolves which network problem, and an interface to the correspondent package. When the planner chooses a package of adaptations, it queries the package and obtains the name and the location of adapter with the instruction how to use it. Each package has its own database that contains the necessary description of the adapters from this package. This database is designed together with the adapters by the same designer. The record in the planner database that describes the access to the database for the package is also created by a package designer.

3) **Ordering of adapters in local plans**

Ordering of adapters for local planning uses least-commitment planning. Pre-calculated partially ordered plans are located in the planner's library. The order in the partial order plan comes from our experience with the order of adaptations. The rest of adaptation orders are ambiguous and must be resolve during the planning process using user preferences, network conditions, etc. The ordering of adaptations helps to build

local plans and for centralized planning helps to create the order of adaptations that will sufficiently reduce the branching of the search tree during the plan optimization.

4) Optimization of the local plans

Optimization of local planning is done through a refinement search in the space of plans. The search occurs through the modification of the location of the adaptations and evaluation of this modification using an evaluation function. The evaluation function calculates a value that corresponds to the quality of a communication and the amount of network resources used. The modification of the location of the adapters occurs through the merging of local plans preserving the previously calculated order of the adapters. The strategy of plan merging is discussed in Section 4.3.

5) Evaluation function

The evaluation function for the optimization search contains the quality of data transfer and resources necessary to run the adaptations. Both these factors are directly dependent; the function should reflect the point of equilibrium between them, which can very for different networks and connections. For our implementation we assume that quality of data transfer is more important than adapter execution resources of nodes. This assumption will be expressed with the correspondent weight coefficients applied to transfer-quality and execution-resources factors. The factors that we are planning to evaluate are:

- Quality of data transfer: throughput, security, reliability, etc.
- The execution resources: CPU cycles, memory

6) Switch between plans

The destination node should be able to switch between locally and centrally calculated global plans for the connection. Each data packet should carry the identification of the plan version that must be applied to the packet. Panda nodes dispatch the packets according to this identification.

7) Extensibility

The system must demonstrate the ability to extend itself when it is necessary to add an adapter, adapt a package, implement a “new” method of adaptation, or present a “new” problem of network communication. Extension of the system should be accompanied

with necessary modifications in databases and minimal changes in the code of the planner.

The result of the implementation is a system to automatically and properly deploy adaptations that help to fix the problems of networks.

4.6 Measurements

The evaluation and the measurement of the work of the planner will be performed on peer-to-peer network connections that consist of one or more links. Multimedia data transfer will be used for the measurements: color images, video, etc.

The preliminary list of the potential network problems and their potential solutions in parenthesis is:

- Throughput (compression)
- Latency (prefetching and caching)
- Burstiness (buffering)
- Security (encryption)

The preliminary list of the potential remedies is:

- Compression (Ziv Lempel, color drop, quality reduction)
- Prefetching and caching: (prefetcher, cache)
- Buffering (buffer)
- Security (public key encryption)

To make the measurements for typical cases several scenarios are possible:

1. Local planning: The connection consists of one problematic link; different kinds of problems cause the deployment of a number of adapters. The goal of the test is to verify the efficiency of the ordering of adapters and the timing of the elementary planning.
2. Centralized planning: The connection consists of two links; each link has its own set of problems. The goal is to verify the efficiency of the planner to calculate an optimized central plan.
3. Centralized planning: The connection consists of a various number links with some "average" number of adapters. The goal of the test is to
    - measure the average "per link" speed of the local planning

- find the dependency of the centralized planning duration on the number of links of the connection

4. Centralized planning: “Realistic case” – some interesting scenario, for example: a mobile computer communicates with a base station, which via telephone line communicates with the Internet. The goal is to make a benchmark for some realistic case.
5. Centralized planning: “Multicast case” – some multicast connection. The goal is to show that even fragmental planning for peer-to-peer fractions of the multicast tree improves the connection.
6. We are planning to deploy Panda with the planner in our office for testing, debugging, more extensive measurements, and casual use.

To demonstrate the advantages of automated planning we will make a comparison of quality of service of a data stream with and without planning for all scenarios. To define a tradeoff between improved quality of service and extra latency that is brought by planning the following measurements of time consumed will be done:

1. On-line planning data gathering
2. Plan calculation:
    1) adapter selection
    2) adapter ordering
    3) optimization of the centralized plan
3. Plan deployment (optionally as Panda designers can already presume this)
4. Switching between plans
5. Plan execution

The improved quality of service can be recognized visually during demonstration, through the actual measuring of the quantity of packets delivered and dropped within a unit of time, or both.

The contemporary implementation of ANs demonstrates higher latency of communications than conventional Internet because of the overhead, Java slowness, etc. The comparison of the ANs equipped with the planner and the conventional networks is problematic.

6. Schedule

To visualize the progress of work involved into the planner implementation we have constructed a preliminary schedule of the research with milestones and estimated completion times.

1. System built
    1. Working framework — Fall 2000
    2. Full scale system (improved framework) — Winter 2001
    3. Tuning the system — Spring 2001
4. Measurements/Demonstration done — Summer 2001
5. Dissertation written — Fall 2001

6. Summary

We showed the current status of planning for open architectures. We presented the most critical problems that planning for ANs faces. The solution of planning problem is somewhere on the cross point between open architecture technology and AI planning theory.

We presented the approach to the planner design for ANs. Panda is the example of an active network system that provides an adaptation service for end-to-end connections and automated planning for the selection and deployment of adaptations. The planning process presumes that a search for a feasible plan in the space of all possible plans will succeed. The complexity of the search depends on the scale of the plan space. We believe that in a practical system the plan space is very large, making automated planning a hard Artificial Intelligence problem.

At the same time, finding a feasible plan is limited by the temporal constraints of a real-time application. Our work is focused on the methods of fast and efficient plan space traversal for possible solutions. We have outlined what we believe to be the key components and tradeoffs to the problem. We are investigating the chosen planning strategy and how well it suits the open architecture we are working with and the kinds of problems we are interested in solving. We presented a number of scenarios of

communication for which we will make controlled measurements of the efficiency of the system.

We are currently working on the planner implementation for use in the Panda prototype. We believe that the planner will improve active network architectures and make active networks better achieve their potential. Legacy applications will benefit from using the planner.

7. References

[Bretthauer95] Kurt M. Bretthauser, Murray J. Côté, "Nonlinear Programming for Multiperiod Capacity Planning in a Manufacturing System", *European Journal of Operational Research*, vol. 96, 1996, pp. 167-179.

[Dean94] Thomas Dean, "Artificial Intelligence. Theory and Practice", The Benjamin/ Cummings Publishing Inc., 1994.

[East 99] East, E.W., "Infrastructure work order planning using genetic algorithms", GECCO-99. *Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA, 1999. P.1510-16.

[Fox98] Armando Fox, Steven D.Griddle, Eric A.Brewer, and Alan Amir, "Adapting to Network and Client Variations Using Infrastructural Proxies: Lessons and Perspectives", *IEEE Personal Communications*, Sept. 1998, 5(4), pp. 10-19.

[Gero98] 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.

[Hauskrecht00] Milos Hauskrecht and Hamish Fraser, "Planning Treatment of Ischemic Heart Disease with partially observable Markov Decision Processes", *Artificial Intelligence in Medicine*, vol. 18, Issue 3, March 2000, pp. 221-244.

[Hicks99] Hicks, M. Keromytis, A.D. (Edited by: Covaci, S.) "A Secure Plan. Active Networks", *First International Working Conference*, IWAN'99. Proceedings, Berlin, Germany: Springer-Verlag, 1999. p.307-14.

[Ihrig96] Laurie H. Ihrig and Subbarao Kamhaampati, "Design and Implementation of a Replay Framework based on the Partial Order Planner", *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Cambridge, MA, 1996. p.849-54.

[Jo98] Jun H. Jo and John S. Gero, "Space Layout Planning Using an Evolutionary Approach", *Artificial Intelligence in Engineering*, vol. 12, 1998, pp. 149-162.

[Joseph95] Anthony D. Joseph, Alan F. deLespinasse, Joshua A. Tauber, David K. Giffort, and M. Frans Kaashoek, "Rover: A Toolkit for Mobile Information Access*", in Proceedings of the Second ACM International Conference on Mobile Computing and Networking (Mobicom '96)*, Nov. 1996.

[Kaebling98] Leslie Pack Kaebling, Michael L. Littman, and Anthony R. Cassandra, "Planning and Acting in partially observable stochastic domains", *Artificial Intelligence*, vol. 101, 1998, pp. 99-134.

[Kambhampati94A] Kambhampati, S., "Refinement search as a unifying framework for analyzing planning algorithms", Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94), Bonn, Germany, 24-27 May 1994.

[Kambhampati94B] Kambhampati, S. and Craig A. Knoblock, "Planning as Refinement Search: A Unified Framework for Evaluating Design Tradeoffs in Partial-Order Planning", submitted to Elsevier Science, 1994.

[Kelly88] Kelly, F. B., "Routing in circuit-switched networks: optimization, shadow prices and decentralization "*Advances in Applied Probability*, vol. 20, 1988, 112-144.

[Liljeberg96] Mika Liljeberg, Heikki Helin, Markku Kojo, and Kimmo Raatikainen, "Enhanced Services for World-Wide Web in Mobile WAN Environment", University of Helsinki, CSD, Report C-1996-28.

[Ling97] Zong Ling, Yun, D.Y.Y, "A planning-based graph matching algorithm for knowledge structure retrieval", *Advances in Concurrent Engineering*, Basel, Switzerland: Technomic Publishing, 1997. p.223-30.

[Mallet97] A. Mallet, J. Chung, and J. Smith, "Operating System Support for Protocol Boosters," HIPPARCH Workshop, June 1997.

[Merigu99] S.Merugu, S.Bhattacharjee, Y.Chae, M.Sanders, K.Calvert and E.Zegura, "Bowman and CANEs: Implementation of an Active Network", presented as an invited paper at the *37th annual Allerton Conference on Communication*, Control and Computing, Monticello, Illinois, September 1999.

[Nilsson98] N.J. Nilsson, "Artificial Intelligence: A New Synthesis", Morgan Kauffman Publishers, San Francisco, California, 1998.

[Noble97] Brian D. Noble, Dushyanth Narayan, James Eric Tilton, Jason Flinn, and Kevin R. Walker, "Agile Application-Aware Adaptation for Mobility", *Proceedings of the 16th ACM Symposium on Operating Principles*, St. Malo, France, October 1997.

[Reiher00] Peter Reiher, Richard Guy, Mark Yarvis, and Alexey Rudenko, "Automated Planning for Open Architectures", Short paper to be presented at *Openarch 2000,* March 2000.

[Russel95] Stuart J. Russel and Peter Norvig "Artificial Intelligence", Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[Tennenhouse97] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden, "A Survey of Active Network Research*", IEEE Communications Magazine,* Jan. 1997, 35(1):80-86.

[Weld94] Deniel S. Weld, "An Introduction to Least Commitment Planning", AI Magazine, vol.15, (no.4), Winter 1994. p.27-61.

[Whetherall98] Wetherall, D.J.; Guttag, J.V.; Tennenhouse, D.L., "ANTS: a toolkit for building and dynamically deploying network protocols*", 1998 IEEE Open Architectures and Network Programming*, San Francisco, CA, 3-4 April 1998, pp.117-29.

[Yarvis99A] Mark Yarvis, An-I A. Wang, Alexey Rudenko, Peter Reiher, and Gerald J. Popek., "Conductor: Distributed Adaptation for Complex Networks", UCLA Tech Report CSD-TR-990042, August 1999.

[Yarvis99B] Mark Yarvis, Peter Reiher, and Gerald J. Popek, "Conductor: A Framework for Distributed Adaptation", *Proc. Seventh Workshop on Hot Topics in Operating Systems (HotOS VII)*, Rio Rico, AZ, March 1999.

[Yarvis00] Mark Yarvis, Peter Reiher, and Gerald J. Popek, "A Reliability Model for Distributed Adaptation", *Openarch 2000*, March 2000.

[Zegura98] AN Composable Services Working Group, "Composable Services for Active Networks", Ellen Zegura, editor, http://www.ittc.ukans.edu/anserve/

[Zhou97] Zhou, G.; Gen, M., **"**Evolutionary computation on multicriteria production process planning problem", *Proceedings of 1997 IEEE International Conference on Evolutionary Computation* (ICEC '97) New York, NY, 1997. p.419-24.