Заключение. Даже некоррелированная случайная дискретная информация при использовании обычных методов сжатия может быть уменьшена по объёму. Для этого можно варьировать как коэффициентами сжимающей матрицы, так и механизмом квантования. Эксперименты показали, что изменение положения строк сжимающей дочерней матрицы не меняет степень сжатия. Более эффективным является вариация величин коэффициентов ДКП. Вариация обычно производится со значениями коэффициентов меньшими единицы. Но это условие не является жёстким – коэффициенты могут быть любыми. Соотношение между величинами коэффициентов ДКП должно изменяться плавно, без скачков. В противном случае эффект сжатия теряется. В экспериментах не удалось строку матрицы вида ($c4$ -$c4$ -$c4$ $c4$ $c4$ -$c4$ -$c4$ $c4$, где $c4$=0.378) заменить более приемлемым эквивалентом. Не удалось пока также объяснить устойчивость этого явления.

В экспериментах менялись значения четырёх коэффициентов матрицы ДКП. Очевидно, более эффективным является вариация семи коэффициентов при оптимизации степени сжатия, но такой поход требует значительных усилий. Необходимо совершенствовать механизм квантования сжимаемых данных.

Литература

1. Эксперименты по сжатию видеоданных / [Кривуца В.Г., Булгач В.Л.,Дикарев А.В., Лазаренко В.Н.] // Вісник Державного університету інформаційно-комунікаційних технологій. – 2012. – Т.10, №3. – С. 5-13.

2. Вариация коэффициентов дискретного косинусного преобразования / [Кривуца В.Г., Булгач В.Л., Дикарев А.В., Лазаренко В.Н.] // Вісник Державного університету інформаційно-комунікаційних технологій. – 2012. – Т.10, №2. – С. 5-10.

УДК 004.67; 004.75

Ткачев В.Н., асп.; **Саваневич В.Е.,** д.т.н. *(Харьковский нац. унив.-т радиоэлектроники)*
Анненков А.Б. *(Запорожский институт экономики и информационных технологий)*
Брюховецкий А.Б., к.т.н. *(Нац. Центр управления и испытаний космических средств)*

МЕТОД ПРЕДОТВРАЩЕНИЯ ВОЗНИКНОВЕНИЯ КОЛЛИЗИЙ ПРИ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКЕ ДАННЫХ В ДЕЦЕНТРАЛИЗОВАННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ

Ткачов В.М., В.Є. Саваневич В.Є, Анненков О.Б., Брюховецький О.Б.. Метод запобігання виникненню колізій при паралельній обробці даних в децентралізованих обчислювальних системах. Запропонований метод запобігання виникненню колізій при паралельній обробці даних в децентралізованих обчислювальних системах, що дозволяє підвищити оперативність обробки даних. Метод реалізований за допомогою повідомлень-запитів та повідомлень-станів.

Ключові слова: КОЛІЗІЯ, ПАРАЛЕЛЬНА ОБРОБКА ДАНИХ, ОБЧИСЛЮВАЛЬНА СИСТЕМА

Ткачев В.Н., Саваневич В.Е., Анненков А.Б., Брюховецкий А.Б. Метод предотвращения возникновения коллизий при параллельной обработке данных в децентрализованных вычислительных системах. Предложенный метод предотвращения возникновения коллизий при параллельной обработке данных в децентрализованных вычислительных системах, что позволяет повысить оперативность обработки данных. Метод реализован с помощью сообщений-запросов и сообщений-состояний.

Ключевые слова: КОЛЛИЗИЯ, ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ, ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА

Tkachov V.M. , Savanevych V.Ie., Annenkov O.B., Brukhovetskiy O.B. Method of prevention of collisions for parallel data processing in decentralized computation systems. Method of prevention of collisions for parallel data processing in decentralized computation systems, allowing to improve efficiency of data processing. The method is implemented using the request-messages and status-message.

Key words: COLLISIONS, PARALLEL DATA PROCESSING, COMPUTATION SYSTEMS

Введение. Децентрализованные вычислительные системы с параллельной обработкой данных появились во второй половине 80-х годов [1].

Параллельная обработка данных позволила повысить оперативность обработки больших массивов данных и эффективность использования вычислительной системы [2]. Это актуально, например, для обработки астрономических данных в распределенной вычислительной системе. Однако, из-за возможности обращения нескольких экземпляров программы к одной и той же порции данных с целью дальнейшей ее обработки (ситуация коллизии) методы параллельной обработки данных являются малоэффективными. При этом снижается как оперативность обработки данных, так и достоверность результатов многоэтапного процесса обработки. Поэтому, целесообразна разработка эффективного метода предотвращения возникновения коллизий при параллельной обработке данных.

Анализ литературы. Подходы к предотвращению коллизий в различных операционных системах сложилась на фоне развития сетевых технологий. Первые методы предотвращения коллизий были основаны на принципах организации технологии Ethernet. Так, в технологии Ethernet, чтобы корректно обработать коллизию, все вычислительные станции одновременно наблюдают за возникающими в разделяемой среде сигналами. Если передаваемые и наблюдаемые сигналы отличаются, то фиксируется обнаружение коллизии. Для увеличения вероятности скорейшего обнаружения коллизии всеми станциями сети станция, которая обнаружила коллизию, прерывает передачу своего кадра (в произвольном месте, возможно, и не на границе байта) и усиливает ситуацию коллизии посылкой в сеть специальной последовательности из 32 бит, называемой jam-последовательностью [3]. Однако, практически это давало незначительный выигрыш по сравнению с существующим методом планирования обработки данных, когда коллизию можно было избежать за счет связки «экземпляр программы – массив обрабатываемых данных».

В UNIX системах использовался метод блокировок [4]. Для получения блокировки обрабатываемой порции данных экземпляром программы создается уникальный временный файл, полное имя которого содержит в себе его идентификатор процесса (или комбинацию идентификаторов процесса и потока, осуществляется блокировка между отдельными потоками). Затем экземпляром программы вызывается функция link для создания ссылки на этот файл с определенным заранее именем. После успешного создания сам файл может быть удален экземпляром программы вызовом функции unlink. После осуществления работы с блокировкой файл с известным именем удаляется экземпляром программы командой unlink. Если link возвращает ошибку EEXIST, экземпляр программы должен попытаться создать ссылку еще раз. Основные недостатки этого метода сводятся к тому, что если экземпляр программы, установивший блокировку, прекратит работу досрочно, не сняв ее, порция данных не будет обработана. Этот метод также несовершенен, поскольку идентификатор может быть использован повторно. Условия, при которых этот метод функционирует, не позволяют его использовать в сетях с Windows-ориентированными ОС.

В операционных системах семейства Windows для предотвращения коллизиям ОС накладывает "замки" (lock) на порции данных [5]: замок разделяемого доступа для порции данных, открытого для чтения, и замок монопольного доступа для порции данных, открытого для записи. Замок может входить в дескриптор открытой порции данных. Если новый замок конфликтует с замком, наложенным ранее начавшейся операцией, то новая операция должна быть заблокирована. Большинство Windows-ОС обеспечивает только разделение доступа к каждой отдельной порции данных, перепоручая управление сложными транзакциями и целостностью порций данных дополнительному программному обеспечению – менеджерам транзакций (IBM CICS, MS TransactionServer и др.). Общим недостатком вышеуказанных методов является необходимость изменять структуру поступающих порций данных и отсутствие мультиплатформенности.

Децентрализованная распределённая параллельная вычислительная система. Среди разновидностей вычислительных систем была выбрана децентрализованная

распределённая параллельная вычислительная система. Одним из основных преимуществ распределённой параллельной вычислительной системы является возможность использования более мощных вычислительных ресурсов, чем имеются в наличии локально. В децентрализованной вычислительной системе решение об обработке данных принимается самостоятельно каждым экземпляром программы на основании текущего состояния порции данных или серии порций данных. Одним из основных атрибутов таких систем является общее виртуальное хранилище данных, что позволяет одновременное выполнение всех задач, поставленных перед вычислительной системой.

Децентрализованная распределенная параллельная вычислительная система состоит из вычислительных станций μ_i $(i=\overline{1,N_\mu})$. На i-ой вычислительной станции μ_i функционирует ξ_{ki} $(k=\overline{1,N_\xi})$ экземпляров программ, равное или меньшее количеству ядер процессоров станции. Цикл обработки данных состоит из N_j этапов. В ℓ-ых хранилища данных на j-ом этапе обработки $(\alpha_{j\ell}$ $(\ell=\overline{1,N_\alpha}))$ хранятся порции данных с идентификаторами n с порядковыми номерами m из серии ω:

$$\psi_{j\ell nm\omega}\left(\begin{cases} j=\overline{0,N_j} \\ \ell=\overline{0,N_\alpha} \\ \omega=\overline{1,N_\omega} \\ n=\overline{1,N_n} \end{cases}\right)$$

Постановка задачи. Порции данных от источника поступают во входной виртуальный буфер децентрализованной вычислительной системы. Источником может быть любая система хранения данных, телескоп, виртуальная обсерватория, пользователи [6, 7]. Порции данных могут поступать по приоритету (по вызову) и в общей очереди. Считается, что входные данные могут иметь принадлежность к любому из этапов общего цикла обработки данных. Вычислительная система, представляющая собой совокупность вычислительных станций, на которых функционируют экземпляры программы, к моменту начала обработки порций входных данных получает команду (сообщение) о начале работы.

Количество экземпляров программы равно количеству ядер процессоров вычислительных станций вычислительной системы. Каждый экземпляр программы в любой момент времени может выполнять любой из этапов обработки данных.

Цель статьи. Необходимо, разработать метод предотвращения коллизий при параллельной обработке данных в децентрализованных вычислительных системах. Разрабатываемый метод не должен нарушать целостность входных порций данных.

Система сообщений-запросов и сообщений-состояний. В основе разрабатываемого метода лежит использование сообщений-запросов и сообщений-состояний:

– β_{jikn} сообщение-запрос о n-ой порции данных на j-ом этапе обработки k-го экземпляра программы, который работает на i-ой вычислительной станции;

– $\beta_{jik\omega}$ сообщение-запрос о ω-ой серии порции данных на j-ом этапе обработки k-ым экземпляром программы, который работает на i-ой вычислительной станции;

– $\beta_{j\ell\omega e}$ сообщение-состояние ω-ой серии порций данных в ℓ-ом хранилище порций данных на j-ом этапе обработки:

$e=1$ – сообщение-состояние формирования серии порций данных;

$e=2$ – сообщение-состояние сформированной серии порций данных, готовых к обработке;

$e=3$ – сообщение-состояние обработки серии порций данных;

$e=4$ – сообщение-состояние обработанной серии порций данных.

При этом совокупности однотипных сообщений во время использования хранятся в очередях сообщений:

– γ_j очередь сообщений-запросов на j-ом этапе обработки серий и порций данных;

– γ_{je} очередь e-сообщений-состояний на j-ом этапе обработки серий порций данных.

Экземпляр программы ξ_{ki} $(k=\overline{1,N_\xi})$ состоит из ядра (core), компонентов и плагинов (plugins) [8].

Описание вычислительного метода предотвращения коллизии при параллельной обработке данных. Для каждой порции данных вычислительный метод предотвращения коллизии при параллельной обработке данных заключается в следующем.

1. На нулевом ($j=0$) этапе обработки, функционирующий на i-ой вычислительной станции k-ый экземпляр программы ξ_{ki} $(k=\overline{1,N_\xi})$ ищет во входном буфере $\alpha_{j\ell}$ $(j=0,\ell=0)\to(\alpha_{jl}=\alpha_{00})$ очередную m-ую порцию данных $\Psi_{j\ell nm\omega}$ $((j=0,\ell=0)\to\Psi_{00nm\omega}$ ω-ой серии с идентификатором n.

2. При отсутствии во входном буфере α_{00} порции данных $\Psi_{00nm\omega}$, экземпляр программы ξ_{ki} выполняет поисковый цикл сообщений состояния $\beta_{j\ell\omega e}$ по ℓ-ым хранилищам порций данных на всех этапах обработки.

2.1. При обнаружении экземпляром программы ξ_{ki} в ℓ-ом хранилище порций данных сообщения состояния $\beta_{j\ell\omega e}$, которое позволяет проводить обработку серий ($\beta_{j\ell\omega 2}$ – сообщение состояния о сформированной и готовой к обработке ω-ой серии порций данных), он ищет сообщение-запрос $\beta_{jik\omega}$ на обработку этой серии порций данных.

2.1.1. При отсутствии сообщения-запроса $\beta_{jik\omega}$ о ω-ой серии порции данных, переход на пункт 8.2.

2.1.2. При наличии сообщения-запроса $\beta_{jik\omega}$ о ω-ой серии порции данных, переход на пункт 1.

2.2. При обнаружении экземпляром программы ξ_{ki} в ℓ-ом хранилище порций данных сообщения состояния $\beta_{j\ell\omega e}$, которое не позволяет проводить обработку серий, он продолжает поиск в $(\ell+1)$-ом хранилище порций данных.

2.2.1. При отсутствии задач (во всех ℓ-ых хранилищах порций данных серии формируются, обрабатываются или уже обработаны) k-ый экземпляр программы делает уведомление внешней программе о переходе в режим работы «Ожидание».

2.2.2. При поступлении уведомления от внешней программы о выходе из режима «Ожидание» до истечения времени $\tau_{\max}$, экземпляр программы переходит к пункту 1.

2.2.3. При отсутствии уведомления от внешней программы о выходе из режима «Ожидание» по истечению времени $\tau_{\max}$, экземпляр программы завершает работу.

3. При наличии во входном буфере α_{00} порции данных $\Psi_{00nm\omega}$, k-ый экземпляр программы ξ_{ki} ищет в очереди сообщений-запросов γ_0 $((j=0)\to\gamma_0)$ сообщение-запрос β_{00kin} о порции данных с идентификатором n.

3.1. При наличии сообщения-запроса β_{00kin} о порции данных, переход к пункту 1.

3.2. При отсутствии сообщения-запроса β_{00kin} о порции данных, k-ый экземпляр программы ξ_{ki}, создает в очереди сообщений-запросов γ_0 k-ое сообщение-запрос β_{00kin}.

4. Функционирующий на i-ой вычислительной станции, k-ый экземпляр программы ξ_{ki}, ожидает наперед заданное время τ_0.

5. Экземпляр программы ξ_{ki} ищет в очереди сообщений-запросов γ_0 сообщение-запрос β_{00kin} о порции данных.

5.1. При обнаружении k-ым экземпляром программы ξ_{ki} нескольких сообщений-запросов $\beta_{00kin}\,(k=\overline{1,N_k})$, данная порция данных будет перемещена этим экземпляром программы при условии, что k-ый номер ее сообщения запроса меньше k'-ых номеров других сообщений-запросов $(k'>k)$.

5.2. При тождестве k и k'-ых $(k'\equiv k)$ сообщений-запросов $\beta_{jik\omega}$, порция данных будет перемещена k-ым экземпляром программы ξ_{ki} при условии, что номер вычислительной станции i, на которой он функционирует, меньше номеров i' других вычислительных станций $(i'>i)$.

5.3. Остальные экземпляры программы ξ_{ki}, переходят к пункту 1.

6. Функционирующий на i-ой вычислительной станции, k-ый экземпляр программы перемещает m-ую порцию данных $\Psi_{00nm\omega}$ ω-ой серии с идентификатором n из входного буфера α_{00} в хранилище порций данных $\alpha_{j\ell}\,((\ell=1,j=1)\rightarrow(\alpha_{j\ell}=\alpha_{11})$ для первого этапа обработки.

7. Экземпляр программы ξ_{ki} создает сообщение состояния о формировании ω-ой серии порций данных ($\beta_{j\ell\omega1}$) с целью предотвращения обработки несформированной ω-ой серии другим экземпляром программы.

7.1. Экземпляр программы ξ_{ki} проверяет каждую входящую m-ую порцию данных $\Psi_{11nm\omega}$ на предмет выявления последней из ω-ой серии.

7.1.1. При обнаружении не последней порции данных $\Psi_{11nm\omega}$ из ω-ой серии, экземпляр программы ξ_{ki} переходит к пункту 1.

7.1.2. При обнаружении последней порции данных $\Psi_{11nm\omega}$ из ω-ой серии, или, если серия содержит единственную порцию данных, экземпляр программы ξ_{ki} создает сообщение состояния о сформированной ω-ой серии порций данных $\beta_{j\ell\omega2}$.

7.2. Экземпляр программы ξ_{ki} удаляет сообщение состояния о формировании ω-ой серии порций данных $\beta_{j\ell\omega1}$.

8. Экземпляр программы ξ_{ki} ищет в очереди сообщений-запросов γ_j сообщение-запрос $\beta_{jik\omega}$ о ω-ой серии порций данных.

8.1. При наличии сообщения-запроса $\beta_{jik\omega}$ о серии порций данных, переход к пункту 1.

8.2. При отсутствии сообщения-запроса $\beta_{jik\omega}$, k-ый экземпляр программы ξ_{ki}, создает в очереди сообщений-запросов γ_j k-ое сообщение-запрос $\beta_{jik\omega}$.

9. Функционирующий на i-ой вычислительной станции, k-ый экземпляр программы ξ_{ki}, ожидает наперед заданное время τ_0.

10. Экземпляр программы ξ_{ki} ищет в очереди сообщений-запросов γ_j сообщение-запрос $\beta_{jik\omega}$ о ω-ой серии порции данных.

10.1. При обнаружении k-ым экземпляром программы ξ_{ki} нескольких сообщений-запросов $\beta_{jik\omega}\,(k=\overline{1,N_k})$, данная серия порций данных будет обрабатываться этим

экземпляром программы при условии, что k-ый номер ее сообщения запроса меньше k'-ых номеров других сообщений-запросов $(k' > k)$.

10.2. При тождестве k и k'-ых $(k' \equiv k)$ сообщений-запросов $\beta_{jik\omega}$, серия порций данных будет обрабатываться k-ым экземпляром программы ξ_{ki} при условии, что номер вычислительной станции i, на которой он функционирует, меньше номеров i' других вычислительных станций $(i' > i)$.

10.3. Остальные экземпляры программы ξ_{ki}, переходят к пункту 1.

11. Функционирующий на i-ой вычислительной станции, k-ый экземпляр программы осуществляет j-ый этап обработки ω-ой серии порций данных $\Psi_{j\ell nm\omega}$.

11.1. Перед началом обработки ω-ой серии порций данных экземпляр программы ξ_{ki}, создает в очереди сообщений состояния γ_{je} сообщение состояние о процессе обработки данной серии порций данных $\beta_{j\ell\omega 3}$. Это предотвращает обращения k'-го экземпляра программы с целью обработки этой же серии порций данных.

11.2. После окончания обработки ω-ой серии порций данных экземпляр программы ξ_{ki} создает в очереди сообщений состояния γ_{je} сообщение об окончании обработки этой серии порций данных $\beta_{j\ell\omega 4}$. Это предотвращает обращения k'-го экземпляра программы с целью повторной обработки этой же серии порций данных.

11.3. Экземпляр программы ξ_{ki} удаляет сообщение состояние о процессе обработки ω-ой серии порций данных $\beta_{j\ell\omega 3}$.

12. Функционирующий на i-ой вычислительной станции, k-ый экземпляр программы ξ_{ki} выполняет проверку j-го этапа обработки на предмет определения его последнего из количества этапов обработки порций данных N_j.

12.1. При определении не последнего этапа обработки, экземпляр программы ξ_{ki} переходит к пункту 7.

12.2. При определении последнего этапа обработки, экземпляр программы ξ_{ki} формирует отчет-уведомление о завершении обработки серии порций данных на последнем этапе и переходит к пункту 1.

Результаты эксперимента. Экспериментальные исследования проводились на российском дистанционно управляемом телескопе автоматизированной обсерватории ISON-NM [9]. Вычислительная система состояла с одной 4-ядерной вычислительной станции.

Результаты обработки серии кадров в обсерватории ISON-NM были доступны через 20 минут после окончания формирования последнего кадра серии. Это дает выигрыш во времени приоритета открытия объекта примерно в 10% случаев (50-100 зарегистрированных в МРС предварительных открытий в год), что немаловажно в условиях сложившейся конкурентной борьбы между обсерваториями мира [9].

Выводы. В статье разработан метод предотвращения коллизий при параллельной обработке данных. Метод отличается от известных тем, что роль блокирующих механизмов выполняют сообщения-запросы и сообщения-состояния, в общем виде представлены файлами; обеспечивается поддержка масштабируемости вычислительной системы.

Предложенный метод позволяет повысить показатели оперативности при обработке данных. Метод апробирован на российском дистанционно управляемом телескопе автоматизированной обсерватории ISON-NM. Метод может быть применен также в других системах параллельной обработки данных, где задействовано больше одного экземпляра программы, то есть вычислительная система состоит как минимум с одной вычислительной станции с 2-ядерным процессором.

Дальнейшие работы над методом целесообразно посвятить исследованию его работы на предмет выявления ошибок в больших распределенных вычислительных системах.

Литература

1. Титоренко Г.А. Автоматизированные информационные технологии в экономике / Г.А. Титоренко. – М.: Компьютер, ЮНИТИ, 1998. – 400 с.

2. Воеводин В. В. Параллельные вычисления / В. В. Воеводин, Вл. В. Воеводин. –СПб.: БХВ-Петербург, 2002. – 608 с.

3. Олифер В. Г. Компьютерные сети. Принципы технологии, протоколы / В. Г. Олифер, Н. А. Олифер. – [3-е изд.]. – СПб.: Питер, 2010. – 943 с.

4. Стивенс У.Р. Unix. Взаимодействие процессов / У.Р. Стивенс. – СПб.: Питер, 2006. – 576 с.

5. Деревянко А.С. Операционные системы / А.С. Деревянко, М.Н. Солощук. –Харьков: НТУ "ХПИ", 2002. – 574 с.

6. Leon Alexander, Adrie Stander, Jacques Ophoff. Using a network telescope to analyse Internet traffic for network for ensicreadiness // Annual Conference on World Wide Web Applications. – 2012.

7. Peter Fox. The Role of Virtual Observatories and Data Frameworks in an Era of Big Data. – NIST bIGdATA. – Gaithersburg, MD. – 2012.

8. Гагарина Л. Г.Технология разработки программного обеспечения / Л. Г. Гагарина, Е.В. Кокорева, Б. Д. Виснадул. – М.: ИД «ФОРУМ»; ИНФРА-М, 2008. – 402 с.

9. Еще раз об OLDAS [Електронний ресурс] // – Режим доступу: http://neoastrasoft.com/news/information_about_oldas/?lang=ru (06.02.2012).

УДК 621.394/.396.019.3

Маліна Т.І., нач. відділу *(ТОВ «Астеліт»),* **Колченко Т.В.,** пров. спец. *(Держспецзв'язку)*

ВІДНОВЛЕННЯ ПРАЦЕЗДАТНОСТІ ТЕЛЕКОМУНІКАЦІЙНОЇ МЕРЕЖІ ОПЕРАТОРА ТЕЛЕКОМУНІКАЦІЙ: ВНУТРІШНІ І ЗОВНІШНІ РЕСУРСИ

Маліна Т.І., Колченко Т.В. Відновлення працездатності телекомунікаційної мережі оператора телекомунікацій: внутрішні і зовнішні ресурси. Визначено ресурси, які необхідні для відновлення працездатності телекомунікаційної мережі.

***Ключові слова**:*, ТЕЛЕКОМУНІКАЦІЙНА МЕРЕЖА, ВІДНОВЛЕННЯ ПРАЦЕЗДАТНОСТІ

Малина Т.І., Колченко Т.В. Восстановление работоспособности телекоммуникационной сети оператора телекоммуникаций: внутренние и внешние ресурсы. Определены ресурсы, необходимые для восстановления функционирования телекоммуникационной сети.

***Ключевые слова**:* ТЕЛЕКОММУНИКАЦИОННАЯ СЕТЬ, ВОССТАНОВЛЕНИЕ РАБОТОСПОСОБНОСТИ

Malina T.I., Kolchenko T.V. Internal and external resources, necessary for proceeding in the capacity of telecommunication network of telecommunications operator. Certainly resources which are needed for proceeding in the capacity of telecommunication network.

***Keywords**:* TELECOMMUNICATION NETWORK, PROCEEDING CAPACITY

Надзвичайні ситуації, значні аварії та катастрофи техногенного характеру впливають на життя і здоров'я населення, середовище, а також на діяльність і функціонування телекомунікаційних мереж. При цьому основні завдання захисту – виявлення характеру надзвичайних ситуацій і можливостей зниження ризиків, зменшення наслідків аварій і забезпечення безперервного функціонування телекомунікаційних мереж та їх відновлення – повинні вирішуватися безпосередньо на окремих підприємствах.